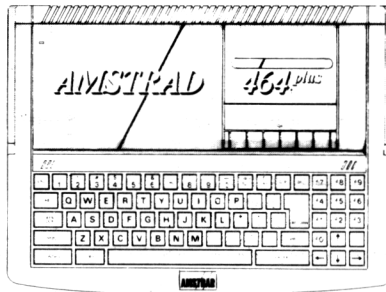
464 PLUS

6128 PLUS

AMSTRAD

464/6128 Plus
Manuale d'uso

Ogni operazione di manutenzione e servizio deve essere effettuata a cura dei rivenditori autorizzati AMSTRAD. AMSTRAD non accetterà responsabilità per nessuna perdita o danno causati da operazioni di manutenzione o servizio effettuate da personale non autorizzato. Questo manuale intende semplicemente assistere il lettore all'uso del prodotto e quindi AMSTRAD non può essere ritenuta responsabile per eventuali perdite o danni derivanti dall'uso di informazioni o di errori o di omissioni contenuti in questo manuale o da un uso non corretto del prodotto.

Prima pubblicazione 1990

Edizione inglese a cura di Roland Perry

Pubblicato da AMSTRAD SpA

PRINTED IN ENGLAND

Il prodotto descritto in questo manuale ed i prodotti da utilizzare con esso sono soggetti ad un continuo sviluppo e miglioramento.

Questo manuale viene fornito gratuitamente ed intende assistere il lettore nell'uso del prodotto. Le informazioni contenute in questo manuale e la relativa documentazione vengono forniti in buona fede da AMSTRAD.

AMSTRAD NON GARANTISCE L'ACCURATEZZA DELLE INFORMAZIONI E NON SI ASSUME RESPONSABILITA' E NON EFFETTUERA' RISARCIMENTI PER EVENTUALI PERDITE O DANNI SUBITI A CAUSA DELL'USO DI QUALUNQUE INFORMAZIONI FORNITA O EMESSA.

Questo manuale può contenere informazioni tecniche imprecise o errori tipografici. Alle informazioni contenute in questo manuale vengono apportate continue modifiche; queste ultime vengono incluse nelle nuove pubblicazioni. AMSTRAD può migliorare e/o modificare il prodotto e i programmi descritti nel manuale in qualsiasi momento senza avvisare gli utenti di tali modifiche.

Il prodotto descritto in questo manuale non è stato progettato e non deve essere usato in attività e con funzioni critiche nelle quali un errore o un calo di tensione possa danneggiare cose o persone.

Occorre seguire attentamente le istruzioni contenute nella garanzia fornita con il prodotto e compilarla.

IMPORTANTE

I termini e le istruzioni contenuti nella garanzia fornita con il prodotto devono essere seguiti con cura.

ESCLUSIONI DI DANNI CONSEQUENZIALI

IN NESSUN CASO, AMSTRAD ACCETTERA' DOMANDE DI RISARCIMENTO PER UN QUALUNQUE DANNO O PERDITA SUBITI A CAUSA DELL'USO O DI UN GUASTO DEL PRODOTTO O DI QUALUNQUE INFORMAZIONE FORNITA, INCLUSO MA NON LIMITATO A DANNI ECONOMICI O FINANZIARI, DANNI A APPARECCHI PERIFERICI O PRODOTTI, PERDITE DI USO, PRODUTTIVITA' E TEMPO.

IMPORTANTE

Leggere prima dell'uso ...

Note relative all'installazione

1. Si colleghi il cavo di alimentazione sempre alla presa a 3 pin seguendo le istruzioni contenute nella prima parte intitolata 'Predisposizione'.

2. Non si colleghi mai il sistema ad una fonte d'alimentazione diversa da 220-240V 50Hz.

3. All'interno degli apparecchi non vi sono parti che possano essere manipolate da un utente. Non si cerchi perciò di aprire questi apparecchi. Ogni operazione di servizio dovrà essere effettuata da personale qualificato.

4. Per evitare disturbi agli occhi, il monitor deve essere sistemato il più lontano possibile dalla tastiera ed utilizzato in una stanza adeguatamente illuminata. Il controllo di luminosità del monitor dovrebbe essere tenuto nella posizione più bassa possibile.

5. Non bloccare o coprire i fori per la ventilazione.

6. Non lasciare o usare il sistema in zone eccessivamente calde, fredde, umide o polverose.

7. Non inserire o estrarre la cartuccia quando il computer è acceso.

8. Non utilizzare il computer senza cartuccia.

Compatibilità

Il 464/6128 Plus può essere usato con tutto il software e le periferiche per i CPC464, 664 e 6128. Nonostante tutti gli sforzi per eliminare le incompatibilità è possibile che alcuni di questi possano non operare correttamente. Prima di acquistare programmi o periferiche per i modelli precedenti si consulti il rivenditore.

N.B. I connettori delle periferiche di espansione, delle stampanti e del secondo drive hanno la stessa funzione di quelli dei modelli precedenti ma possono essere differenti.

Note operative (uso dei dischi - 6128)

Non ci si preoccupi se non si comprende appieno il gergo usato in questo paragrafo; l'importanza di queste note diverrà chiara proseguendo nella lettura di questo manuale.

1. Non accendere o spegnere mai il sistema se vi è un disco nel drive. In questo modo è possibile danneggiare il disco e perdere programmi e dati di grande importanza.

2. Fare sempre copie di back-up (duplicati) dei dischi che contengono programmi importanti. E' in particolar modo importante fare copie dei dischi contenenti il sistema CP/M forniti con il 6128. Altrimenti, se si dovessero danneggiare tali dischi, la loro sostituzione potrebbe rivelarsi assaï costosa.

3. Assicurarsi di non sovrascrivere per sbaglio i dischi contenenti il sistema CP/M, mantenendo aperti i fori di protezione dei dischi.

4. Per la piena affidabilità del sistema, il drive non deve essere posto di fronte al monitor ma leggermente a destra. Non porre il sistema vicino ad altre sorgenti di interferenze elettriche.

5. Non toccare mai la superficie del disco all'interno della custodia di plastica.

6. Non estrarre un disco mentre si sta leggendo o scrivendo su di esso.

7. Ricordarsi che la formattazione di un disco cancella tutti i dati contenuti su di esso.

8. I dischi ed i drive devono essere tenuti lontani dai campi magnetici

9. La licenza d'uso dei dischi di sistema del CP/M (che sono numerati elettronicamente) permette il loro uso su un solo computer. In particolare ciò significa che è proibito dare a qualunque altra persona un disco di CP/M con il proprio numero codificato. Si legga attentamente la licenza d'uso del programma per l'utente finale riportata nell'Appendice 1 verso la fine del manuale.

Indice

Capitolo 7
Ulteriori informazioni su Bank Manager

Capitolo 8
A vostra disposizione

Appendici

CAPITOLO 1
CORSO INTRODUTTIVO

Parte 1: Predisposizione ...

Il personal computer 464/6128 può essere collegato ad uno dei seguenti dispositivi:

1. Un monitor monocromatico MM12 AMSTRAD
2. Un monitor a colori CM14 AMSTRAD

Come inserire la spina di alimentazione

Il 464/6128 funziona con una tensione di alimentazione di 220-240 Volt, 50 Hz C.A..

Importante

I fili del cavo di alimentazione sono colorati secondo il seguente schema:

verde e giallo:	terra
blu:	neutro
marrone:	fase

Dal momento che i colori dei fili del cavo di alimentazione potrebbero non corrispondere ai segni colorati sulla spina, si proceda come segue:

Il filo colorato in verde e giallo va connesso al terminale della spina contraddistinto con la lettera E o dal simbolo di terra o colorato in verde o verde e giallo.

Il filo colorato in blu deve essere collegato al capo contraddistinto con la lettera "N" o colorato in nero.

Il filo colorato in marrone deve essere collegato al capo contraddistinto con la lettera "L" o colorato in rosso.

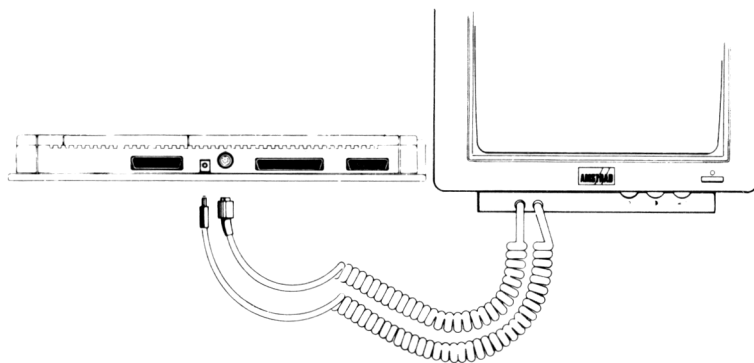
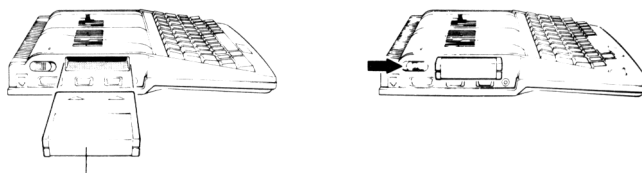
Quando il sistema non è in uso si estragga la spina dalla presa.

Non occorre effettuare connessioni interne. Non si cerchi dunque di accedere ai componenti interni del computer. Attenersi sempre all'etichetta di avvertenza posta sotto al computer e sul retro del monitor:

ATTENZIONE! ALTA TENSIONE. NON RIMUOVERE ALCUNA VITE

Predisposizione del 464/6128

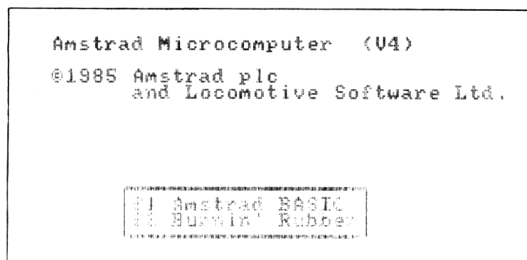
IMPORTANTE: Scegliendo la posizione del computer ci si assicuri che:

- si trovi vicino alle prese di alimentazione
- non si trovi vicino a sorgenti di calore (ad esempio un termosifone), o vicino ad un rubinetto dell'acqua e che non sia esposto alla luce solare diretta. Questi elementi possono infatti danneggiare il computer.

1. Ci si assicuri che il monitor non sia collegato alla presa di alimentazione.
2. Si connetta la presa ad 8 pin situata ad una estremità del cavo che fuoriesce dal monitor con la presa DIN a 8 pin situata sul retro del computer.
3. Si connetta il capo della presa più piccola situata in fronte al monitor nella presa 5V DC sul retro del computer.

4. Inserire la cartuccia fornita nella presa posta sul lato sinistro del computer. IMPORTANTE: la cartuccia può essere inserita o estratta solo se l'interruttore di accensione è in posizione OFF.

5. Inserire la spina nella presa di alimentazione.

6. Premere il tasto Power sul pannello frontale del monitor. Quando questo tasto non è premuto anche l'unità di sistema non è alimentata.

7. Accendere il computer mediante l'interruttore posto sul lato sinistro.

Se tutte le connessioni sono esatte verrà visualizzata l'immagine seguente:

Per evitare un affaticamento degli occhi, si regolino i controlli di luminosità ☼ e di contrasto ◑ posti sul pannello frontale del monitor. E' inoltre presente il controllo di volume ◁.

Come scegliere tra BASIC e gioco dimostrativo

La cartuccia fornita con il 464/6128 contiene sia il BASIC AMSTRAD che un gioco dimostrativo. Per scegliere il BASIC AMSTRAD si prema il tasto **[F1]** sul lato destro della tastiera entro 30 secondi dall'accensione del computer. Se invece si preme il tasto **[F2]** o non si preme il tasto **[F1]** entro 30 secondi dall'accensione, verrà automaticamente caricato il gioco dimostrativo.

Per ulteriori informazioni sull'uso del BASIC AMSTRAD si consulti il manuale.

N.B. Per giocare con il gioco dimostrativo dopo aver usato il BASIC AMSTRAD, immettere il comando **|GAME** e premere il tasto **[ENTER]**. Il simbolo | viene prodotto premendo contemporaneamente il tasto **[SHIFT]** ed il tasto con i simboli @ e |.

Parte 2: Connessione di periferiche ...

Questo paragrafo spiega come connettere diverse periferiche al 464/6128. Le spiegazioni ed i dettagli dell'uso di queste periferiche si troveranno all'interno del manuale nei paragrafi specifici. Nella parte 9 del Capitolo 6 vi sono alcune figure che riportano le varie porte presenti sul 464/6128.

IMPORTANTE: NON CONNETTERE O SCONNETTERE LE PERIFERICHE SE IL COMPUTER E' ACCESO.

Joystick e Paddle controller (PD-1)

Con il 464/6128 viene fornito un Paddle controller. E' inoltre possibile collegare ad esso un secondo joystick o un Paddle controller che potrà essere utilizzato solo con giochi che ne permettono l'uso.

Connettere la spina del Paddle controller alla presa joystick più vicina alla parte frontale posta sul lato sinistro del computer. Il secondo joystick o Paddle controller dovrà essere collegato alla presa joystick più vicina alla parte posteriore.

Per ulteriori informazioni sulle scritture di programmi che utilizzano i joystick si prosegua nella lettura del manuale.

Penna ottica/Light gun

La penna ottica e la Light gun possono essere utilizzate solo con giochi che ne prevedono l'uso.

Collegare la penna ottica o la Light gun alla presa AUX sul lato sinistro del computer.

Questi oggetti possono essere utilizzati solo con software su disco, cassetta o cartuccia.

Joystick analogico

Il joystick analogico può essere utilizzato solo con giochi che ne prevedono l'uso. Il joystick analogico permette un controllo molto più preciso di quello possibile con un joystick o un Games controller.

Collegare il joystick analogico alla presa AUX sul lato sinistro del computer.

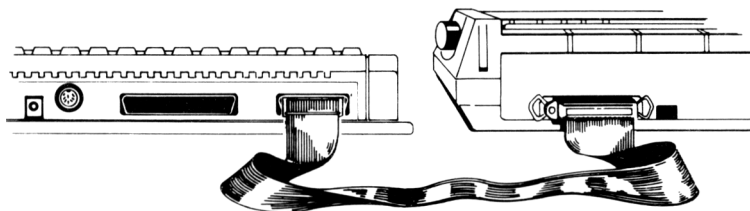
Questo oggetto può essere utilizzato solo con software su disco, cassetta o cartuccia.

Stampante

Il 464/6128 può essere connesso ad una qualsiasi stampante parallela compatibile Centronics. Se si intende connettere una stampante ci si dovrà procurare un cavo adatto che viene normalmente fornito con quest'ultima.

Collegare una estremità del cavo (25 pin) alla presa PRINTER situata sul retro del computer. L'altra estremità (36 pin) dovrà essere collegata alla corrispondente presa sulla stampante. Se il connettore della stampante possiede dei gancetti di sicurezza sulla presa questi possono essere chiusi.

NON collegare il connettore Centronics a 36 pin alla presa a 36 pin situata sul retro del computer: tale presa serve infatti a collegare un secondo disco.

Per ulteriori informazioni sulle operazioni di stampa si consulti il resto del manuale.

Amplificatore/altoparlanti esterni

Il 464/6128 può essere connesso ad un amplificatore stereo. Il cavo di ingresso dell'amplificatore stereo deve terminare con uno spinotto jack di 3,5mm che deve essere inserito nella presa STEREO del computer.

Il computer fornirà in uscita un segnale a tensione fissa e perciò si dovranno usare i controlli dell'amplificatore stesso per regolare il volume, il bilanciamento ed il tono.

Espansione

E' possibile connettere al 464/6128 periferiche tipo interfacce seriali, modem, penne ottiche, ROM, ecc. mediante la presa EXPANSION situata sul retro del computer.

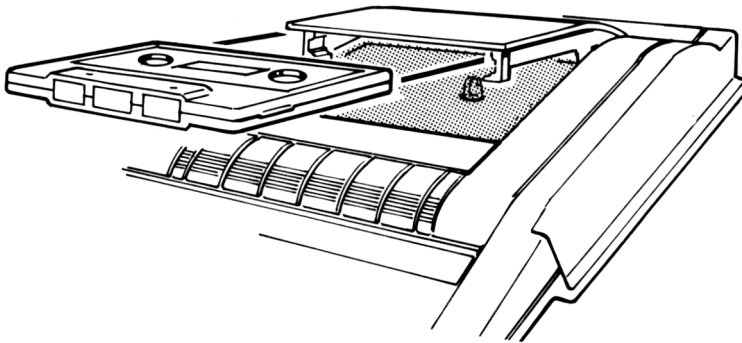
Secondo drive (solo 6128)

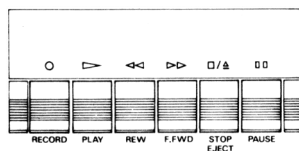
Sull'unità di sistema è previsto un connettore per il collegamento di un secondo disco. Per ulteriori informazioni consultare la documentazione allegata al drive.

Parte 3: Dischi e cassette ...

La memoria del 464/6128 è in grado di mantenere un programma memorizzato solo fino a quando il computer è connesso alla corrente e l'unità stessa è accesa: nella terminologia informatica tale memoria è detta "volatile". Se si vogliono memorizzare i programmi o i dati quando la corrente è spenta occorre memorizzarli su cassetta o su disco.

Controlli delle cassette (solo 464)

Sulla destra della tastierà vi è il registratore. La meccanica di tale registratore è essenzialmente equivalente ad un sistema audio ad eccezione del fatto che il controllo del segnale elettronico è ottimizzato per l'uso di memorizzazione di dati su un computer.

Il modo corretto di inserire una cassetta nel registratore

Analogamente le operazioni della tastierina del registratore sono analoghe a quelle della maggior parte dei registratori audio.

I controlli del registratore del 464

Si noti che tali tasti vanno premuti più forte rispetto ai tasti di una macchina per scrivere.

[REC] (opera simultaneamente al tasto **[PLAY]** per registrare i dati). Non è possibile attivarlo a meno che non venga inserita una cassetta con la linguetta di 'protezione da scrittura' (si veda la figura della pagina seguente) e non venga chiuso lo sportellino. Per attivare la funzione si deve mantenere premuto il tasto **[REC]** e premere il tasto **[PLAY]**. Il computer scriverà i dati su cassetta quando gli verrà indicato dal programma attualmente in memoria o mediante un comando SAVE inserito da tastiera.

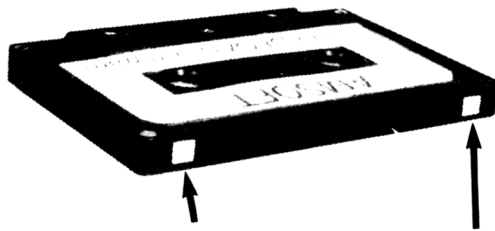
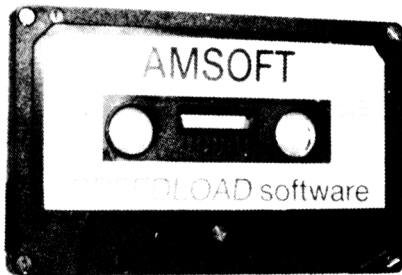
[PLAY] Aziona il meccanismo di trascinamento del nastro per caricare (LOAD) o eseguire (RUN) un programma. Il computer leggerà i dati dalla cassetta, quando verrà indicato dal programma residente in memoria oppure mediante un comando inserito da tastiera. Sia il tasto **[REC]** che **[PLAY]** verranno rilasciati quando il nastro raggiunge la fine.

[REW] Riavvolge il nastro dalla bobina destra verso quella sinistra. Non esiste un meccanismo di cancellazione di questa funzione al termine del riavvolgimento del nastro; non si deve quindi lasciare girare il registratore altrimenti la trasmissione del motorino del nastro potrebbe surriscaldarsi quando il nastro si ferma.

[F.F] Fast Forward ovvero avanzamento veloce: farà avanzare il nastro velocemente dalla bobina sinistra verso quella destra. Non esiste un meccanismo di cancellazione di questa funzione al termine del riavvolgimento del nastro; non si deve quindi lasciare girare il registratore altrimenti la trasmissione del motorino del nastro potrebbe surriscaldarsi quandoil nastro si ferma.

[STOP/EJECT] Ferma qualsiasi operazione sulla cassetta e riporta tutti i tasti della tastierina del registratore nella posizione normale. Se questo tasto viene rilasciato e successivamente premuto ancora lo sportellino del registratore si aprirà permettendo di estrarre la cassetta o di inserirne una. La cassetta non può essere estratta fino a quando l'operazione del drive non è cessata.

[PAUSE] Il meccanismo di pausa opera insieme al tasto **PLAY** o ai tasti **[REC][PLAY].** Non deve essere usato durante una lettura o una registrazione di dati in caso contrario verrà indicato un messaggio di errore. Tutte le operazioni di pausa durante la lettura e la registrazione sono gestite dalle istruzioni interne del software del CPC464, e quindi tale comando sarà poco usato.

Protezione dalla scrittura

LINGUETTE DI PROTEZIONE
DALLA SCRITTURA

Tutte le cassette sono dotate di una linghetta di protezione che serve a prevenire cancellazioni involontarie dei dati presenti su esse. Questa linguetta può essere rimossa facendola saltare via mediante un piccolo paio di pinze o un oggetto analogo: in tal modo non si sarà in grado di tener premuto il tasto **[REC]** poichè la cassetta è così protetta dalla scrittura.

Le possibilità di protezione dalla scrittura sono applicabili ad OGNI LATO della cassetta; quindi per proteggere entrambi i lati della cassetta da scrittura occorre rimuovere entrambe le linguette. Nel caso si voglia successivamente ripristinare la possibilità di scrittura sulla cassetta occorre applicare, sul foro che si è creato dopo l'eliminazione della linguetta, del nastro adesivo.

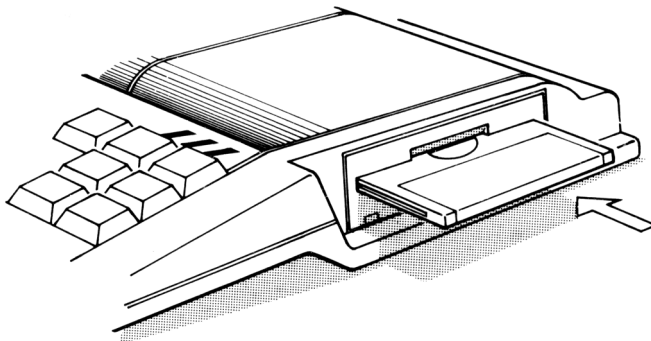
Caricamento di cassette

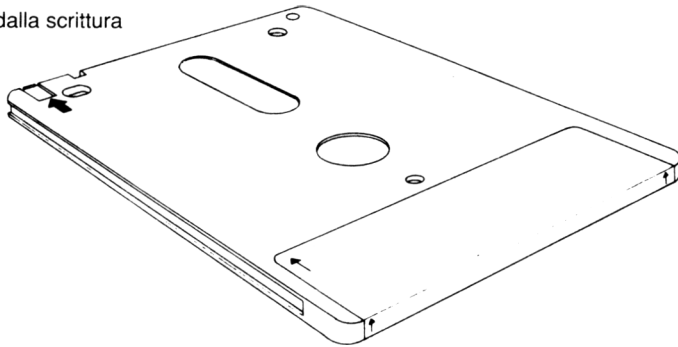
Il nastro della cassetta deve essere completamente riavvolto (dalla bobina destra a quella sinistra): se così non fosse, si prema il tasto **[REW]** fino a quando il nastro viene riavvolto. Se il nastro è uscito dall'apertura della cassetta, prima di inserire quest'ultima nel registratore, occorrerà riportare il nastro all'interno di essa: è comunque possibile in questo modo aver perso delle informazioni memorizzate.

Si noti che mentre è possibile usare nastri che hanno risentito di varie forme di cattivo uso, principalmente pieghe, e altri danni alla superficie del nastro per registrazioni audio non è possibile trattare una registrazione di dati di un computer nello stesso modo ed aspettarsi ancora che continui ad operare in modo affidabile.

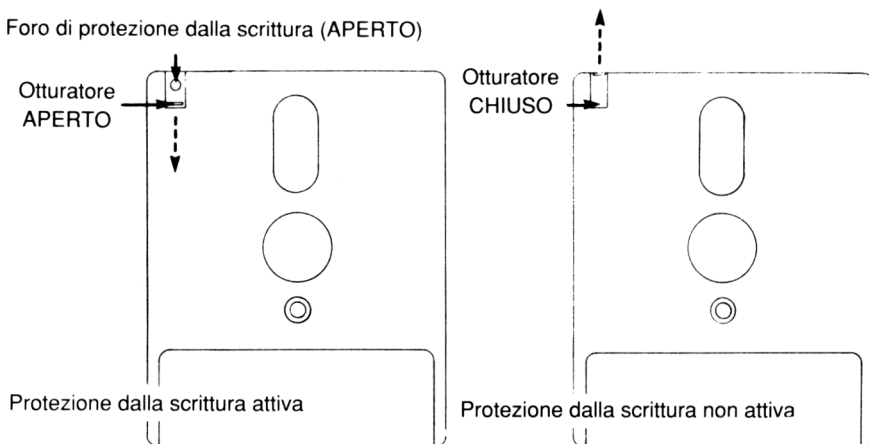
Se si danneggia il nastro ma si è ancora in grado di caricare o eseguire un programma, si deve immediatamente salvare il programma su un nuovo nastro (mentre il programma si trova ancora nella memoria del 464) e gettare la cassetta in modo che non venga la tentazione di usarla di nuovo.

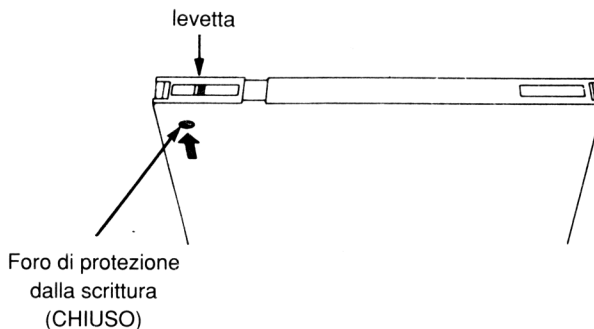
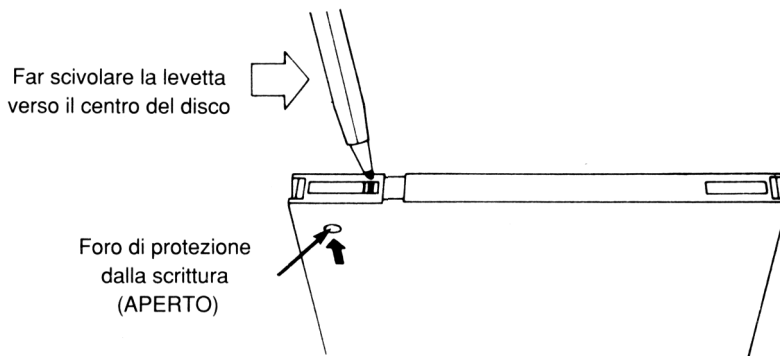
Inserimento dei dischi (solo 6128)

Ogni lato del disco può essere usato separatamente. Il disco deve essere inserito all'interno del drive con il lato contenente l'etichetta mantenuto all'esterno e con quest'ultima rivolta verso l'alto:

Protezione da scrittura

Nell'angolo in alto a sinistra di ogni lato di un dischetto si potrà vedere una piccola freccia che indica un foro coperto da un quadrato di plastica. Questo è il foro di protezione dalla scrittura e serve a proteggere da cancellazioni o 'riscritture' involontarie:

Quando tale foro è chiuso i dati possono essere 'scritti' dal computer sul disco. Nel caso in cui tale foro sia aperto il disco non permette che i dati vengano scritti: in tal modo si potranno evitare cancellazioni involontarie di programmi importanti.

I dischi di diverse case produttrici potrebbero essere dotati di meccanismi di protezione dalla scrittura diversi. Le operazioni da effettuare sui normali dischetti sono:

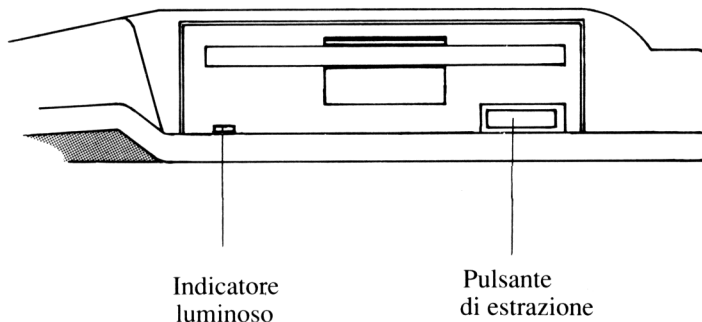
Per aprire il foro di protezione dalla scrittura si faccia scivolare verso il basso il piccolo quadrato di plastica situato nell'angolo in alto a sinistra del disco; il foro sarà aperto.

Per chiudere il foro di protezione dalla scrittura si faccia scivolare il quadrato di plastica in senso contrario (fino a quando il foro non è chiuso).

Altri dischetti usano una piccola leva di plastica situata in una guida nell'angolo in alto a sinistra:

Per aprire una protezione su un tale tipo di disco si faccia scivolare la levetta verso il centro del disco usando la punta di una penna o un oggetto analogo:

Si noti che in qualunque metodo di protezione dalla scrittura l'apertura del foro implica la protezione da scrittura.

IMPORTANTE

Ci si assicuri sempre che il foro di protezione di scrittura dei dischetti di sistema CP/M sia aperto.

Quando i dischi sono inseriti

Nel pannello frontale del drive del 6128 vi è un indicatore luminoso rosso ed un pulsante di estrazione:

Indicatore luminoso

Indica che il computer sta leggendo o scrivendo dati sul disco.

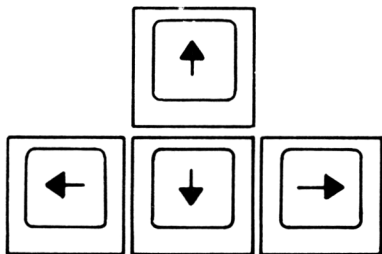
Se il secondo drive è connesso, l'indicatore luminoso di quest'ultimo (Drive B) sarà costantemente acceso. Esso si spegnerà quando il drive principale all'interno del computer (Drive A) sta leggendo o scrivendo su disco.

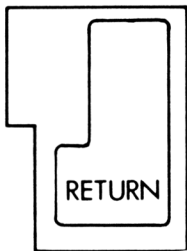
Pulsante di estrazione

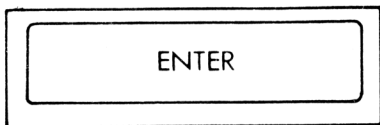
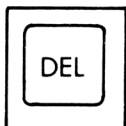
La pressione di tale pulsante fa sì che sia possibile estrarre il disco dal drive.

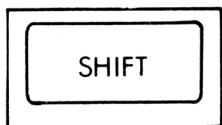
Parte 4: Iniziamo ...

Prima di iniziare a caricare e salvare programmi su disco o cassette è bene familiarizzare con alcuni tasti presenti sul computer. Gli utenti già esperti possono saltare la lettura di questo paragrafo.

Con il computer acceso e con il messaggio iniziale presente sullo schermo proviamo ad usare diversi tasti ...

I tasti cursore ⇑ ⇓ ⇐ ⇒ (nell'angolo in alto a destra della tastiera) spostano il cursore (il piccolo quadrato) nello schermo.

Premendo uno alla volta ciascun tasto si vedrà il cursore muoversi nello schermo.

Il tasto **[RETURN]** inserisce le informazioni scritte nel computer. Dopo aver premuto tale tasto si inizia una nuova linea. Ogni istruzione inserita deve essere seguita dalla pressione da parte dell'utente del tasto **[RETURN]**.

Da questo momento in poi, ogni volta che verrà indicato **[RETURN]** si intenderà la pressione di tale tasto dopo ogni istruzione o linea di programma.

In circostanze normali (per difetto) questo tasto ha lo stesso effetto del tasto **[RETURN]** e può essere quindi usato in luogo di questo. Comunque, così come i tasti del tastierino numerico anche il tasto **[ENTER]** può essere ri-definito per altri usi. Ciò verrà spiegato successivamente in maggior dettaglio in questo manuale.

Questo tasto è usato per cancellare sullo schermo il carattere a sinistra del cursore (ad esempio una lettera o un numero).

Si scriva a b c d; si potrà notare che la lettera d è situata alla destra del cursore. Se si vuole cancellare tale lettera si prema una volta il tasto **[DEL]**: la lettera d verrà cancellata. Se si continua a mantenere premuto tale tasto anche le lettere a b e c verranno cancellate.

Vi sono due tasti **[SHIFT]**. Se si preme uno dei due e lo si mantiene premuto mentre si preme un tasto sullo schermo apparirà una lettera maiuscola o il segno più in alto su un tasto che ne contiene due.

Si scriva la lettera e poi si mantenga premuto il tasto **[SHIFT]** e si scriva nuovamente tale lettera. Sullo schermo apparirà:

e E

Si scrivano ora alcuni spazi premendo la barra spazi. Si provi ad usare uno dei tasti contrassegnati da numeri della prima linea della tastiera. Si scriva il numero 2 poi, mantenendo premuto il tasto **[SHIFT]** si scriva di nuovo tale tasto. Sullo schermo si vedrà:

2"

Si provi a premere uno qualunque dei tasti da solo o con il tasto **[SHIFT]**.

Questo tasto ha una azione simile al tasto **[SHIFT]** ma questo va premuto una sola volta. Con tale tasto premuto tutte le lettere inserite verranno scritte in maiuscolo mentre i tasti contrassegnati con numeri verranno scritti uguali (non viene cioè scritto il segno situato sopra il numero).

Si prema **[CAPS LOCK]** e si scriva:

abcdef1234567

sullo schermo si vedrà:

ABCDEF1234567

Si sarà notato che tutte le lettere sono state scritte in maiuscolo mentre invece di scrivere i simboli sopra i numeri sono stati scritti questi ultimi. Nel caso si vogliano scrivere i simboli sopra i numeri mentre è premuto il tasto **[CAPS LOCK]**, occorre premere il tasto **[SHIFT]**. Si provi a scrivere la seguente sequenza mantenendo premuto il tasto **[SHIFT]**:

abcdef123456

Sullo schermo si vedrà:

ABCDEF!"#$%&

Per tornare in modalità minuscola si prema una volta il tasto **[CAPS LOCK]**.

Se si desiderano scrivere tutti i tasti in maiuscolo e nel contempo anche i simboli situati sopra i numeri senza dover mantenere costantemente premuto il tasto **[SHIFT]** è possibile premere insieme:

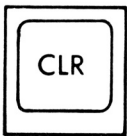
Mantenendo premuto il tasto **[CONTROL]** si prema una volta il tasto **[CAPS LOCK]**. Si scriva ora:

abcdef123456

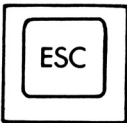
Sullo schermo si vedrà:

ABCDEF!"#$%&

Si noti che, mentre è attivo **[CONTROL] [CAPS LOCK]** è comunque possibile scrivere i numeri usando i tasti funzione (da f0 a f9) situati sulla destra della tastiera.

Mantenendo premuto il tasto **[CONTROL]** e premendo **[CAPS LOCK]** si ritorna nella precedente modalità (ovvero lettere minuscole o solo lettere maiuscole). Se si è tornati in modalità solo lettere maiuscole si prema una volta **[CAPS LOCK]** per tornare in modalità minuscola.

Questo tasto serve a cancellare il carattere su cui è posizionato il cursore.

Si scriva A B C D E F G H. Il cursore sarà posizionato alla sinistra dell'ultimo carattere immesso (H). Si prema ora il tasto freccia a sinistra quattro volte. Il cursore sarà ora posizionato sulla lettera E.

Si noti che la lettera E è comunque visibile. Si prema il tasto **[CLR]** una volta: si vedrà che la lettera E è stata cancellata, le lettere F G H si sono spostate di uno spazio a sinistra e la lettera F è ora posizionata dentro il cursore. Si prema ora il tasto **[CLR]** e lo si mantenga premuto. Si vedrà che la lettera F e successivamente la G e la H verranno cancellate.

Questo tasto è usato per abbandonare una funzione che il computer sta eseguendo. Premendo una volta il tasto **[ESC]** si provocherà una momentanea sospenzione di tale funzione la quale verrà ripresa quando si premerà un qualsiasi tasto.

Premendo invece tale tasto due volte si provocherà l'interruzione definitiva della funzione in corso. Il computer è a questo punto pronto a ricevere ulteriori istruzioni.

Importante

Quando si raggiunge il lato destro dello schermo inserendo su una linea 40 caratteri, il carattere successivo apparirà automaticamente sul lato sinistro della linea successiva. Ciò significa che **NON** si deve necessariamente premere **[RETURN]** quando si raggiunge la fine della linea, come si dovrebbe fare invece su una macchina da scrivere.

Il computer va a capo automaticamente, quindi nel caso si includa un **[RETURN]** non necessario esso stamperà un messaggio di errore (solitamente Syntax error) in fase di esecuzione del programma.

Errori di sintassi

Se sullo schermo appare il messaggio: "Syntax error"significa che il computer non comprende una istruzione inserita. Ad esempio, si inserisca:

printt **[RETURN]**

Nello schermo apparirà il messaggio:

Syntax error

Il messaggio appare perchè il computer non ha compreso l'istruzione printt. Se si commette un errore in una linea di programma, come in:

10 printt "abc" **[RETURN]**

Il messaggio Syntax error non apparirà finchè l'istruzione non verrà esaminata dal computer durante l'esecuzione del programma.
Si immetta:

run **[RETURN]**

(Questo comando indica al computer di eseguire il programma inserito in memoria).
Sullo schermo apparirà:

Syntax error in 10
10 printt "abc"

Questo messaggio indica la linea in cui è capitato l'errore e visualizza la linea di programma ed il cursore per permettere di correggerla. Si prema il tasto "cursore a destra" [->] fino a portare il cursore sulla lettera t, quindi si prema il tasto **[CLR]** per cancellare la lettera ed il tasto **[RETURN]** per immettere la linea corretta.
Si immetta:

run **[RETURN]**

... e si noterà che il computer ha accettato la linea ed ha stampato abc

Parte 5: Caricamento di software e giochi...

Benvenuti a tutti coloro che hanno saltato la parte precedente!

Giochi su cartuccia

Come si è visto nella Parte 1, è possibile eseguire il gioco dimostrativo fornito con il 464/6128 semplicemente premendo il tasto **[F2]**. Se si acquistano delle cartucce non occorre seguire questa procedura; il gioco verrà avviato automaticamente con l'accensione del computer.

Caricamento del software da disco (solo 6128)

Ogni disco dovrebbe contenere le istruzioni per il caricamento del software in esso contenuto; è molto probabile che si debba scrivere qualcosa di simile a:

run "disc" **[RETURN]**

Dopo pochi secondi il gioco verrà caricato in memoria e sarà quindi possibile giocare.

Se il programma non è stato caricato proviamo ad analizzare il messaggio che appare sullo schermo:

Drive A: disc missing
Retry, Ignore or Cancel?

... non si è inserito correttamente il disco o, nel caso si posseggano due drive, lo si è inserito nel Drive B.

DISC not found

...si è inserito il disco sbagliato (o il lato sbagliato del disco) o non si è scritto in modo esatto il nome, DISC

Bad command

...probabilmente si è sbagliato a scrivere DISC inserendo uno spazio o un simbolo di punteggiatura.

Type mismatch

... si sono omesse le virgolette "

Syntax error

... si è sbagliato a scrivere la parola run

Drive A:read fail
Retry, Ignore o Cancel?

... il computer non è riuscito a leggere i dati dal disco. Si controlli di aver inserito correttamente il disco e si prema **R** per Riprovare (Retry). Questo messaggio apparirà nuovamente nel caso si sia danneggiato il disco accendendo o spegnendo il computer con tale disco inserito nel drive.

Una volta appreso come effettuare copie di riserva dei dischi è buona norma farle per tutti i programmi ed in modo particolare per i dischi di sistema di CP/M.

Caricamento del software da nastro (solo 464)

Ogni nastro dovrebbe contenere le istruzioni per il caricamento del software in esso contenuto; probabilmente comunque si dovrà scrivere qualcosa di simile:

run " **[RETURN]**

Esiste anche un modo più veloce per caricare un programma premendo solo due tasti.

1) Mantenendo premuto il tasto **[CONTROL]** si prema il tasto **[ENTER]**

2) Rilasciare i tasti e premere **[RETURN]**

Verrà visualizzato il messaggio:

Press PLAY then any key:

Si dovrà quindi premere il tasto **[PLAY]** posto davanti al registratore. Lo si prema in modo resti premuto. La parte del messaggio 'press any key' (si prema un tasto qualsiasi) è comunemente usata ma può sviare un incauto. E' infatti usata per semplificare le sequenze che indicano al programma di proseguire con l'operazione successiva evitando quindi di specificare ulteriori parole chiave.

Sarebbe stato più corretto dire di premere un tasto qualsiasi ad eccezione dei tasti **[SHIFT][CAPS LOCK][CTRL][ESC]** o uno dei tasti situati sul pannello del registratore ma tutti i computer devono assumere alcune abbreviazioni nell'interesse della semplicità. Quando in questo manuale (o nei programmi che si acquistano) viene indicato 'press any key' si assumono le eccezioni prima sottolineate.

Il tasto che verrà premuto non sarà un carattere visualizzato sullo schermoe servirà solo ad azionare il motorino di trascinamento del nastro. Se il nastro non inizia a girare si prema un altro tasto (è buona abitudine premere il tasto **[ENTER]** più grande) e si controlli che non si sia premuto per sbaglio il tasto **[PAUSE]**.

Una volta che il computer ha iniziato a caricare il programma qualsiasi tasto si prema verrà ignorato.

Si noti che non si è specificato alcun nome di programma. Se dopo l'istruzione

RUN"

non viene specificato alcun nome, il computer cercherà il primo programma che trova sul nastro e lo carica. Quando il computer ha trovato il primo programma registrato correttamente su nastro visualizza sullo schermo il seguente messaggio:

Loading GAMEPLAY 1 block 1

Ciò indica che il computer ha trovato la prima serie di blocchi del programma il cui nome è 'GAMEPLAY'. Ogni programma viene salvato su nastro sotto forma di blocchi di dati (circa 2Kbytes) i quali vengono successivamente letti dal computer: ogni blocco di dati è identificato separatamente sul nastro e il messaggio sullo schermo indica il blocco che il computer sta attualmente leggendo. Dopo ogni blocco di dati il nastro si ferma momentaneamente, poi riprende a girare dopo aver aggiornato il messaggio presente sullo schermo.

Se il computer trova dei dati danneggiati visualizza un messaggio di errore che indica la natura dell'errore. L'unica operazione possibile è quella di cercare di eseguire ugualmente il programma fino a quando non si otterranno più errori.

Assumendo che il nastro sia stato caricato, si leggano le istruzioni sullo schermo, Welcome farà il resto ...

Errori di lettura

Se il messaggio Read error (errore di lettura) appare quando il CPC464 sta cercando di caricare un programma o dei dati dalla cassetta, il nastro continuerà a girare ed il computer continuerà a leggere i blocchi che trova dopo l'errore ma non cercherà di caricarli salvo che essi siano stati identificati come blocco 1 del programma che si è cercato di caricare (senza successo).

Ciò significa che dopo un messaggio di errore di lettura è possibile fermare il nastro usando il tasto **[STOP/EJECT]**, e riavvolgere il nastro (con il tasto **[REW]**) all'inizio e premere di nuovo **[PLAY]**. Il computer avrà in tal modo un'altra possibilità di caricare il programma in cui ha trovato l'errore e, con un po' di fortuna, potrebbe aver successo.

Gli errori di lettura possono sorgere per diverse cause, la più comune delle quali, consiste nel danneggiamento del nastro che può essere piegato, o stirato o avere la superficie rovinata. Tali cause possono anche sorgere quando si spegne il computer con la cassetta inserita ed il tasto **[PLAY]** o i tasti **[REC]** e**[PLAY]** abbassati.

Questo perchè quando il tasto è premuto il nastro è ha contatto della testina e quindi è possibile che un breve impulso elettrico possa passare. Anche se il nastro è fermo, questo impulso, che avviene allo spegnimento (o alla accensione) del computer, può effettivamente danneggiare l'informazione contenuta in quella parte di nastro e renderlo non leggibile. Inoltre, il nastro fermo è chiuso ermeticamente tra il traferro della testina ed il rullo di trazione e quindi se il nastro viene mantenuto a lungo in tale posizione si può piegare.

Gli errori possono anche insorgere a volte per ragioni puramente arbitrarie. Le cassette non sono state originariamente progettate come mezzo di memorizzazione dati ed è per questo che sono imperfette rispetto ai più complessi e più costosi sistemi di memorizzazione 'professionali'.

Gli errori di lettura possono anche sorgere durante la momentanea sospensione (**[PAUSE]**) del processo di registrazione o di lettura o quando la registrazione è stata effettuata con un 464 le cui testine non erano perfettamente allineate.

Ciò nonostante, le cassette hanno eseguito un lavoro eccellente fornendo un 'mezzo' standard a basso costo per memorizzare e reperire informazioni di dati su un computer. La limitazione di carattere fisico della dimensione delle particelle magnetiche sulla superficie del nastro associato alla velocità con cui il nastro passa sotto la testina pone dei limiti alla velocità con cui i dati vengono trasferiti dal nastro al computer. Il tentativo di aumentare la velocità oltre quella tollerata dal sistema porterà ad una operazione poco affidabile: in modo particolare con cassette di software a basso costo.

NOTA: Le cassette contenenti programmi di altri tipi di computer non possono essere lette o caricate dal 464. Anche se possono sembrare uguali (potranno emettere lo stesso tipo di suoni se riprodotte su un impianto audio) non potranno essere né caricate né eseguite.

Parte 6: Programmiamo ...

Fino ad ora abbiamo visto cosa è possibile e cosa non è possibile fare con il computer e come predisporre e connettere le periferiche. Sappiamo come operano alcuni tasti e come caricare programmi. Guardiamo ora qualche istruzione e scriviamole per vedere cosa accade ...

Come tutti, il computer può comprendere solo le istruzioni in un linguaggio che conosce: tale linguaggio è il BASIC (abbreviazione di Beginners' All-purpose Symbolic Instruction Code). Le parole di un vocabolario di BASIC sono dette 'parole chiave' ed ognuna di queste indica al computer di eseguire una particolare funzione. Tutti i linguaggi devono essere conformi a regole grammaticali e BASIC non fa eccezione. In questo contesto la grammatica viene detta 'Sintassi' e il computer sarà sempre pronto ad indicare che si è fatto un Syntax error (errore di sintassi)!

Introduzione alle parole chiave del BASIC AMSTRAD.

Nel Capitolo 3 viene riportata una descrizione completa di tutte le parole chiave del BASIC AMSTRAD. In questa parte introduciamo alcune delle parole chiave del BASIC AMSTRAD più usate nella pratica.

CLS

Per cancellare lo schermo si scriva:

cls **[RETURN]**

(cancella schermo). Lo schermo verrà cancellato e nell'angolo in alto a sinistra dello schermo apparirà la parola Ready (Pronto) ed il cursore.

Si noti che è possibile usare per le parole chiave di BASIC sia le minuscole che le maiuscole.

PRINT

Questa parola chiave viene usata quando si vogliono stampare sullo schermo caratteri, parole o cifre. Si immetta la seguente linea:

print "ciao" **[RETURN]**

Sullo schermo apparirà:

ciao

Le doppie virgolette “” vengono usate per indicare al computer ciò che deve essere stampato. ciao apparirà sullo schermo non appena si premerà il tasto **[RETURN]**. Si scriva ora:

cls **[RETURN]**

... per cancellare lo schermo.

RUN

L’esempio precedente mostrava una sola linea di istruzione. Quando comunque si è premuto **[RETURN]** l’istruzione è stata eseguita ma immediatamente dimenticata. E’ possibile memorizzare una serie di istruzioni da eseguire in un ordine stabilito. Ciò si ottiene scrivendo un ‘programma’. L’ordine delle istruzioni di BASIC inserite nel programma è lo stesso di quello visualizzato, ma all’inizio di ogni riga vi è un numero. Se nel programma vi è più di una linea questi numeri indicano al computer in quale ordine si deve eseguire il programma. Quando si preme il tasto **[RETURN]** la linea viene memorizzata dal computer fino all’esecuzione del programma. Si scriva:

10 print “ciao” **[RETURN]**

Si noterà che alla pressione del tasto **[RETURN]** NON verrà stampato ciao sullo schermo, perchè viene inserita nella memoria la linea di programma. Per eseguire tale programma occorre infatti immettere la parola run. Si scriva:

run **[RETURN]**

Ora apparirà la parola ciao sullo schermo. Al posto di print è possibile usare il simbolo ?, come in:

10 ? “ciao” **[RETURN]**

LIST

Dopo che un programma è stato memorizzato, è possibile verificare ciò che è stato inserito guardando il listato del programma. Si scriva:

list **[RETURN]**

Sullo schermo apparirà:

10 PRINT “ciao”

.. che è il programma memorizzato.

Si noterà che la parola PRINT è in maiuscolo. Ciò significa che il computer ha riconosciuto PRINT come parola chiave del BASIC.

Si scriva ora: cls **[RETURN]** per cancellare lo schermo. Sebbene lo schermo venga cancellato, il programma resta nella memoria del computer.

GOTO

La parola chiave GOTO indica al computer di "saltare" da una linea all'altra, per evitare l'esecuzione di un certo numero di linee o per formare un ciclo. Si immetta:

10 print "ciao" **[RETURN]**
20 goto 10 **[RETURN]**

Poi:

run **[RETURN]**

... e la parola ciao verrà scritta ripetutamente sulla sinistra dello schermo. Ciò avviene perchè la linea 20 di tale programma indica al computer di saltare alla linea 10 e di continuare l'esecuzione da quel punto.

Per fermare l'esecuzione del programma, si prema il tasto **[ESC].** Per riprenderla si prema un qualunque altro tasto. Per fermare definitivamente l'esecuzione, si prema due volte il tasto **[ESC]**.

Si scriva:

cls **[RETURN]**

... per cancellare lo schermo.

Per vedere le parole ciao stampate una in fianco all'altra, linea per linea, fino al riempimento completo dello schermo, si utilizzi lo stesso programma ma con un punto e virgola (;) dopo le ultime virgolette ("). Si immetta:

```
10 print "ciao"; [RETURN]
20 goto 10 [RETURN]
run [RETURN]
```

Si sarà notato che il punto e virgola indica al computer di stampare il prossimo carattere immediatamente dopo il precedente (a meno che il gruppo seguente di caratteri sia troppo lungo per starci sulla stessa linea).

Si termini il programma premendo due volte il tasto **[ESC]**. Ora si reimmetta la linea 10 con una virgola (,) al posto del punto e virgola (;).

```
10 print "ciao", [RETURN]
run [RETURN]
```

Si noterà che la virgola indica al computer di stampare il prossimo carattere (o gruppo di caratteri) dopo 13 posizioni dall'inizio del precedente. Questa possibilità è utile per visualizzare informazioni su colonne. Si noti però che se un gruppo di caratteri è più lungo di 12 caratteri, il gruppo successivo verrà stampato dopo altri 13 caratteri.

Questo limite di 13 caratteri è modificabile mediante il comando ZONE che verrà descritto successivamente in questo manuale.

Di nuovo, per uscire dall'esecuzione del programma si prema due volte il tasto **[ESC]**. Per cancellare completamente la memoria del computer, si tengano premuti i tasti **[SHIFT][CTRL]** ed **[ESC]** esattamente in quest'ordine, ed il computer si reinizializzerà.

INPUT

Questo comando serve per far sapere al computer che deve essere immesso qualcosa, ad esempio la risposta ad una domanda. Si immettano le linee:

```
10 input "quanti anni hai"; eta [RETURN]
20 print "sembri più giovane di ";eta;" anni." [RETURN]
run [RETURN]
```

Sullo schermo apparirà:

```
quanti anni hai?
```

Si immetta la propria età e si prema **[RETURN]**. Se la vostra età era di 18 anni, lo schermo mostrerà:

```
sembri più giovane di 18 anni.
```

Questo esempio mostra l'uso del comando input con una variabile numerica. La parola eta è stata messa alla fine della linea 10 e quindi il computer ha associato la parola eta con il numero immesso ed ha poi stampato questo numero al posto della parola eta nella linea 20. Come è stata usata l'espressione eta per una variabile numerica, avremmo potuto anche usare una sola lettera, ad esempio b.

Si riavvii il computer per cancellare la memoria. (Tasti **[CTRL][SHIFT]** e **[ESC]**). Se si fosse voluto effettuare un input di caratteri (lettere o lettere e numeri) si sarebbe dovuto usare un simbolo di dollaro ($) al termine del nome della variabile. Questo tipo di variabile è detta 'stringa variabile'.

Si inserisca il seguente programma: (Nella linea 20 occorrerà inserire uno spazio dopo la o di ciao ed uno prima della i di il)

> 10 input "Qual'è il tuo nome"; nome$ **[RETURN]**
> 20 print "Ciao ";nome$" il mio nome è Arnold" **[RETURN]**
> run **[RETURN]**

Sullo schermo apparirà:

> Qual'è il tuo nome?

Si immetta il proprio nome seguito da **[RETURN]**

Se il nome immesso era Federico, sullo schermo apparirà:

> Ciao Federico, il mio nome è Arnold

Come è stata usata l'espressione nome$ per una variabile stringa, avremmo potuto anche usare una sola lettera, ad esempio a$. Uniamo ora i due esempi per formare un unico programma.

Si reinizializzi il computer premendo **[CTRL][SHIFT]** e **[ESC]** e poi si immettano le seguenti linee:

> 5 cls **[RETURN]**
> 10 input "Come ti chiami"; a$ **[RETURN]**
> 20 input "Quanti anni hai"; b **[RETURN]**
> 30 print "Devo dire ";a$ " che non dimostri"; b"anni" **[RETURN]**
> run **[RETURN]**

In questo programma abbiamo usato 2 variabili: a$ per il nome e b per l'età. Sullo schermo apparirà:

> Come ti chiami?

Ora si inserisca il proprio nome (ad es. Federico) e poi **[RETURN]**. Verrà chiesto:

> Quanti anni hai?

Ora si inserisca la propria età (ad es. 18) e poi **[RETURN]**.

Se il nome inserito era Federico e l'età 18, sullo schermo apparirà:

Devo dire Federico che non dimostri 18 anni

Correzione di un programma

Se una qualunque delle linee del programma era stata immessa non correttamente ed ha prodotto un messaggio Syntax error o un altro messaggio di errore, è possibile correggere la linea invece di riscriverla. Per provare questa operazione, si inseriscano queste linee di programma non corrette:

5 clss **[RETURN]**
10 input "Come t chiami"; a$ **[RETURN]**
20 input "Quanti anni hai"; b **[RETURN]**
30 print "Devo dire";a$ " che non dimostri"; b"anni" **[RETURN]**

Nel programma vi sono 3 errori:

Nella linea 5 abbiamo inserito clss al posto di cls.
Nella linea 10 abbiamo inserito t invece di ti.
Nella linea 30 abbiamo dimenticato uno spazio tra dire e le virgolette (").

Vi sono 3 modi per correggere un programma. Il primo consiste nella ribattitura della linea. Quando una linea viene ribattuta e immessa, essa sostituisce la linea con lo stesso numero presente nella memoria del computer.
Il secondo metodo usa la parola edit ed infine vi è il metodo cursore copia.

Metodo Edit

Per correggere l'errore alla linea 5,

si immetta:

edit 5 **[RETURN]**

La linea 5 verrà stampata dopo la linea 30 con il cursore posizionato sulla c di clss.

Per cancellare la s in più di clss, si prema il tasto di cursore a destra (->) fino a portare il cursore sull'ultima lettera s e poi si prema il tasto **[CLR].** La s sparirà.

Ora si prema **[RETURN]** e la linea 5 sarà corretta anche in memoria. Si scriva:

list **[RETURN]**

... e si vedrà che la linea 5 è stata corretta.
Il comando **AUTO**, descritto successivamente in questo manuale, può essere usato per correggere diverse linee in modo analogo a quanto descritto per il Metodo Edit.

Metodo Cursore Copia

Il cursore copia è un altro cursore (in aggiunta a quello già presente sullo schermo) che viene visualizzato quando si mantiene premuto il tasto **[SHIFT]** e si preme uno dei tasti cursore. Questo è separato dal cursore principale e può essere spostato in modo indipendente dall'altro.

Per correggere gli errori alle linee 10 e 30, si tenga premuto il tasto **[SHIFT]** e poi si prema il tasto cursore in alto (↑) fino a che il cursore copia si trova all'inizio della linea 10. Si noterà che il cursore principale non si è mosso e che vi sono ora 2 cursori sullo schermo. Ora si prema il tasto **[COPY]** finché il cursore copia si trova nello spazio tra t e chiami. Si noterà che la linea 10 verrà riscritta sull'ultima linea e che il cursore si ferma nella stessa posizione del cursore copia. Ora si batta la lettera i. Essa apparirà solo nell'ultima linea.

Il cursore principale si è spostato mentre il cursore copia è rimasto dove era. Ora si prema il tasto **[COPY]** finché tutta la linea 10 non sarà stampata. Si prema **[RETURN]** e la linea 10 verrà memorizzata. Il cursore copia sparirà ed il cursore principale si porterà sulla nuova linea 10. Per correggere il secondo errore, si tenga premuto il tasto **[SHIFT]** e si prema il tasto cursore in alto (↑) per portare il cursore copia all'inizio della linea 30.

Si prema ora il tasto **[COPY]** fino a quando il cursore copia si trova sulle virgolette vicine alla parola dire. Ora si prema una volta la barra spaziatrice. Sulla linea inferiore apparirà uno spazio. Si tenga premuto il tasto **[COPY]** finché il resto della linea 30 non verrà stampato. Poi si prema **[RETURN].**

Si può ora listare il programma per controllare che le correzioni siano state fatte scrivendo:

list **[RETURN]**

NOTA: Per spostare, in fase di edizione, il cursore in maniera più veloce si tenga premuto il tasto **[CONTROL]** e si prema il tasto cursore a destra o il tasto cursore a sinistra.

Ora si reinizializzi il computer premendo i tasti **[CTRL][SHIFT]** e **[ESC]**.

IF

I comandi IF (SE) e THEN (ALLORA) chiedono al computer di verificare una condizione e prendere successivamente una decisione in base al risultato del test. Nell'istruzione, ad esempio:

if 1+1=2 then print “esatto” **[RETURN]**

... il computer verificherà la condizione e prenderà successivamente una decisione.

La parola chiave ELSE può essere usata, in un comando IF THEN, come azione alternativa nel caso la condizione risultasse falsa, ad esempio:

if 1+1=0 then print “esatto” else print “sbagliato” **[RETURN]**

Estenderemo ora il programma visto precedentemente con l’uso dei comandi IF e THEN.

Si inserisca il seguente programma; si noti che vi sono 2 nuovi simboli. < significa “minore di” e si trova vicino alla lettera M, > significa “maggiore di” e si trova in fiando al tasto < (minore di).

```
5 cls [RETURN]
10 input “Come ti chiami”; a$ [RETURN]
20 input “Quanti anni hai”; anni [RETURN]
30 if anni < 13 then 60 [RETURN]
40 if anni < 20 then 70 [RETURN]
50 if anni > 19 then 80 [RETURN]
60 print “Dunque “a$” sei quasi un ragazzo a”anni”anni”:end [RETURN]
70 print “Dunque “a$” sei un ragazzo a”anni”anni”:end [RETURN]
80 print “Dunque “a$” non sei piu’ un ragazzo a”anni”anni” [RETURN]
```

Per controllare che il programma sia corretto, si scriva:

list **[RETURN]**

Ora si scriva:

run **[RETURN]**

Ora si risponda alle domande presentate dal computer e si veda ciò che succede.

Si capirà ora l’effetto dei comandi IF e THEN all’interno di un programma. Abbiamo anche introdotto la parola END alla fine delle linee 60 e 70. La parola chiave END (fine) viene usata per fermare l’esecuzione del programma. Se nella linea 60 non vi fosse stato un END, il programma avrebbe continuato la propria esecuzione con le linee 70 e 80.

Nello stesso modo se non vi fosse stato un END alla linea 70 il programma avrebbe

continuato la propria esecuzione con la linea 80. I due-punti (:) prima della parola END separa quest'ultima dall'istruzione precedente. I due-punti servono per porre due o più istruzioni in una linea di comando. La linea 5 all'inizio del programma serve per cancellare lo schermo. Da questo momento introdurremo sempre tale comando nei programmi per pulire la memoria.

Si riavvii il computer premendo i tasti **[CTRL][SHIFT]** e **[ESC]**.

FOR e NEXT

I comandi FOR e NEXT sono usati per eseguire delle operazioni un numero specifico di volte. Le istruzioni da ripetere devono essere incluse nel ciclo FOR NEXT.

Si scriva:

5 cls **[RETURN]**
10 for a = 1 to 10 **[RETURN]**
20 print "numero operazione";a **[RETURN]**
30 next a **[RETURN]**
run **[RETURN]**

Si potrà notare che l'istruzione della linea 20 viene eseguita 10 volte, così come specificato dal comando FOR della linea 10. Si noti inoltre che il valore della variabile a viene incrementato ad ogni passo del ciclo di 1.
In un ciclo FOR NEXT è anche possibile inserire la parola chiave STEP la quale serve ad incrementare di un dato valore la variabile. Si modifichi, ad esempio, la linea 10 come segue:

10 for a=10 to 50 step 5 **[RETURN]**
run **[RETURN]**

E' possibile anche usare un valore di incremento negativo:

10 for a=100 to 0 step -10 **[RETURN]**
run **[RETURN]**

REM

REM è la abbreviazione di REMark (ovvero nota). Questa istruzione indica al computer di ignorare tutto ciò che segue sulla linea. E' possibile usare REM per ricordare, ad esempio, il titolo di un programma, l'uso di una variabile:

10 REM Uccisione degli invasori **[RETURN]**
20 L=5 :REM numero di vite **[RETURN]**

L'apice ' (che può essere scritto tenendo premuto il tasto **[SHIFT]** e premendo il tasto 7) può essere usato al posto del comando REM. Ad esempio:

10 'Uccisione degli invasori **[RETURN]**
20 L=5 'numero di vite **[RETURN]**

GOSUB

Se all'interno del programma vi è un gruppo di istruzioni che deve essere ripetuto diverse volte, non occorre scriverlo ripetutamente tutte le volte che occorre; può essere invece inserito in una 'sotto-routine' che viene richiamata tutte le volte richieste mediante il comando GOSUB seguito dal numero di linea. La fine della routine GOSUB è contrassegnata da una istruzione RETURN. A quel punto il computer tornerà ad eseguire l'istruzione seguente il GOSUB.

Nel seguente programma ad esempio:

```
10 a=2
20 PRINT "questa è la tabellina del numero";a
30 FOR b=1 TO 12
40 c=a*b
50 PRINT a;"x";b;"=";c
60 NEXT
70 PRINT
80 '
90  a=5
100 PRINT "questa è la tabellina del numero";a
110 FOR b=1 TO 12
120 c=a*b
130 PRINT a;"x";b;"=";c
140 NEXT
150 PRINT
160 '
170 a=8
180 PRINT "questa è la tabellina del numero";a
190 FOR b=1 TO 12
200 c=a*b
210 PRINT a;"x";b;"=";c
220 NEXT
230 PRINT
240 '
250 a=9
260 PRINT "questa è la tabellina del numero";a
270 FOR b=1 TO 12
```

```
280 c=a*b
290 PRINT a;"x";b;"=";c
300 NEXT
310 PRINT
```

... è possibile vedere i numeri di linea che sono stati ripetuti nei vari punti del programma, ad esempio, il gruppo dalla linea 260 alla linea 310. Possiamo prendere questo gruppo di linee e formare una sotto-routine ed aggiungere una istruzione RETURN alla fine. Successivamente richiameremo tale sotto-routine con una istruzione GOSUB 260 ogni volta che sarà necessario. Il programma risulterà come segue:

```
10 a=2
15 GOSUB 260
80 '
90 a=5
95 gosub 260
160 '
170 a=8
175 gosub 260
240 '
250 a=9
255 gosub 260
256 END
257 '
260 PRINT "questa è la tabellina del numero";a
270 FOR b=1 TO 12
280 c=a*b
290 PRINT a;"x";b;"=";c
300 NEXT
310 PRINT
315 RETURN
```

Le sotto-routine sono una parte principale della programmazione. Esse comportano programmi strutturati e sviluppano ottimi ambienti di programmazione.

Occorre sempre tenere in mente che quando si scrive una sotto-routine non è necessario 'saltare' sempre nello stesso punto. Una sotto-routine scritta dalla linea 500 fino alla linea 800 può essere richiamata come: GOSUB 500, o GOSUB 640 o GOSUB 790.

Si noti che nel precedente programma l'istruzione END è stata posta nella linea 256. Se così non fosse stato il programma sarebbe andato avanti ad eseguire, dopo la linea 255 la linea 260 la quale NON era richiesta da nessuna istruzione GOSUB.

ARITMETICA SEMPLICE

Il computer può anche essere facilmente usato come calcolatrice

Per comprendere ciò si vedano gli esempi seguenti. In questo paragrafo useremo il simbolo ? al posto della parola chiave PRINT. Le risposte verranno stampate subito dopo aver premuto il tasto **[RETURN]**.

ADDIZIONE

(per scrivere il simbolo più si usano i tasti **[SHIFT]** e ;)

Si scriva:

?3+3 **[RETURN]**
6

(NON deve essere inserito il simbolo di uguale (=))

Si scriva:

?8+4 **[RETURN]**
12

SOTTRAZIONE

(per il meno si usa il tasto con il simbolo di =)

Si scriva:

?4-3 **[RETURN]**
1

Si scriva:

?8-4 **[RETURN]**
4

MOLTIPLICAZIONE

(per il tale simbolo si usa **[SHIFT]** e : in quanto * sta per x)

Si scriva:

?3*3 **[RETURN]**
9

Si scriva:

?8*4 **[RETURN]**
32

DIVISIONE

(per il simbolo di divisione si usa il tasto che riporta anche ? in quanto / sta per :)

Si scriva:

?3/3 **[RETURN]**
1

Si scriva:

?8/4 **[RETURN]**
2

DIVISIONE INTERA

(usa il simbolo di divisione \ senza considerare il resto)

Si scriva:

?10\6 **[RETURN]**
1

Si scriva:

?20\3 **[RETURN]**
6

MODULO

(per ottenere il resto di una divisione intera si usi il comando MOD)

Si scriva:

?10 MOD4 **[RETURN]**
2

Si scriva:

?9 MOD 3 **[RETURN]**
0

RADICE QUADRATA

Per trovare la radice quadrata di un numero si usa sqr ( ). Il numero di cui si vuole la radice quadrata dovrà essere posto tra le due parentesi.

Si scriva:

?sqr(16) **[RETURN]**
4

Si scriva:

```
?sqr(100) [RETURN]
10
```

ESPONENZIALE

(si usi il tasto che riporta il simbolo di £)

L'esponenziale è l'elevamento a potenza di un numero. Ad esempio 3 al quadrato, 3 al cubo, ecc.

Si scriva:

```
?3^3 [RETURN]
27
```

Si scriva:

```
?8^4 [RETURN]
4096
```

RADICE CUBICA

Si può facilmente calcolare la radice cubica di un numero usando un metodo simile al precedente. Per trovare la radice cubica di 27, si scriva:

```
?27↑(1/3) [RETURN]
3
```

Per trovare la radice cubica di 125, si scriva:

```
?125↑(1/3) [RETURN]
5
```

CALCOLI MISTI

(+,-,*,/,MOD,\)

I calcoli misti sono compresi dal computer ma calcolati secondo alcune priorità. La massima priorità è data alla moltiplicazione ed alla divisione, poi all'addizione ed alla sottrazione. Queste priorità si applicano a calcoli contenenti solo queste quattro operazioni.

Se il calcolo era:

3+7-2*7/4

Si potrà pensare che verrà calcolato in questo modo:

3+7-2 *7 /4
= 8 *7 /4
= 56/4
= 14

Mentre verrà calcolato nel seguente modo:

3+7-2*7/4
=3+7-14/4
=3+7-3.5
=10-3.5
=6.5

Si provi ad inserire la seguente istruzione:

?3+7-2*7/4 **[RETURN]**
6.5

E' possibile modificare il modo in cui il computer effettua questo calcolo utilizzando delle parentesi. Il computer effettuerà i calcoli contenuti nelle parentesi prima di passare a quelli fuori dalle parentesi. Si dimostri ciò immettendo il calcolo sopra presentato ma utilizzando le parentesi.

Si scriva:

?(3+7-2)*7/4 **[RETURN]**
14

La priorità di TUTTE le operazioni aritmetiche è la seguente:

↑ Esponenziale
MOD Modulo
\- Meno unitario (indica un numero come negativo)
* e / Moltiplicazione e divisione
\ Divisione intera
\+ e - Addizione e sottrazione

ALTRI ESPONENTI

Se nei propri calcoli si devono usare numeri molto grandi o molto piccoli, è talvolta utile usare la notazione scientifica. La lettera E indica l'elevamento a potenza della base 10 ad un numero. Si può usare la lettera e in maiuscolo o in minuscolo.

Ad esempio 300 è la stessa cosa di $3x10^2$. In notazione scientifica perciò si scriverà 3E2. In modo simile 0.03 è la stessa cosa di $3x10^{-2}$. Si provino i seguenti esempi

Si può scrivere:

?30*10 **[RETURN]**
300

oppure:

?3E1*1E1 **[RETURN]**
300

?300*1000 **[RETURN]** ... oppure ... ?3E3*1E3 **[RETURN]**
3000000

?3000*0.001 [RETURN] ... oppure ... ?3E3*1E-3 [RETURN]
3

Parte 7: Come salvare ...

Ora che si è provato a scrivere qualche programma si vorrà probabilmente sapere come salvarlo. Se si possiede un 464 lo si salverà su cassetta. Se si possiede un 6128 lo si salverà su disco. Successivamente si potrà ricaricare il programma tutte le volte che si vorrà.

Oltre alle ovvie differenze fisiche tra cassetta e disco esistono diverse modalità operative. Un disco appena acquistato non può essere estratto dalla custodia ed usato immediatamente come un nastro. Deve essere prima 'formattato'; questo processo verrà descritto di seguito. Inoltre, mentre in un disco le informazioni vengono memorizzate e ripristinate automaticamente, nel nastro questa operazione deve essere svolta manualmente. Il computer comunque non avvia e fa fermare il nastro automaticamente.

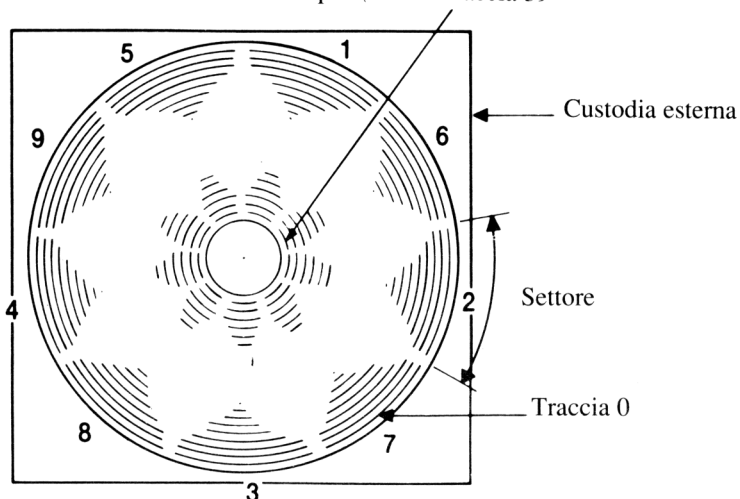
Un altro punto consiste nell'importanza di fornire un 'nome' ai file del disco. I nomi delle cassette seguono generalmente alcuni standard che variano molto e che possono essere omessi. Ciò non è possibile con i dischi. I nomi dei file di disco devono seguire strettamente alcuni standard che verranno descritti successivamente in questo paragrafo.

Se si possiede un 464 si legga il paragrafo 'Come salvare un programma su cassetta'

Formattazione dei dischi

Prima di scrivere dei dati su un disco vergine quest'ultimo deve essere formattato. La formattazione può essere paragonata alla costruzione di una serie di ripiani che vengono distribuiti nel disco per memorizzare informazioni; in altre parole, si pone un ambiente di lavoro organizzato in cui i dati possono essere memorizzati o prelevati.

La formattazione divide il disco in 360 aree separate:

Traccia 39
5
1
Custodia esterna
9
6
4
2
Settore
Traccia 0
8
7
3

Vi sono 40 tracce partendo dall'esterno (Traccia 0) fino all'interno (Traccia 39) del disco ed inoltre la circonferenza di quest'ultimo è suddivisa in 9 settori.

Ogni traccia di un settore può memorizzare 512 byte di dati; quindi il totale disponibile su ogni lato del disco è di 180Kbyte.

Primi passi nell'uso dei dischi di sistema di CP/M

Per predisporre un dischetto vergine affinchè sia possibile leggervi e scrivervi programmi occorre formattarlo. Tale operazione può essere svolta usando il lato 1 (Side 1) del disco di sistema CP/M (fornito insieme al computer).

Si accenda il sistema e si inserisca il lato 1 (Side 1) del disco di sistema nel drive.

Si scriva:

|cpm **[RETURN]**

(E' possibile scrivere il simbolo | mantenendo premuto il tasto **[SHIFT]** e premendo il tasto @).

Dopo pochi secondi in alto sullo schermo si potrà leggere il seguente messaggio:

CP/M Plus for the 6128. (c)1985 Amstrad plc

Questo è un messaggio 'Sign on' che indica che il computer è sotto il controllo del sistema operativo CP/M Plus. Accanto al cursore si potrà vedere sullo schermo anche il simbolo A>. Questo è il prompt (analogo a Ready nelle operazioni BASIC) che indica che il computer è in attesa di istruzioni.

Una volta entrati in ambiente CP/M non è possibile inserire i comandi di BASIC poichè questi non verranno riconosciuti dal sistema operativo.

Se per esempio si inserisce il comando BASIC:

cls **[RETURN]**

Il computer risponderà:

CLS?

...indicando che non ha compreso il comando.

Per vedere brevemente quali sono i comandi di CP/M si scriva:

dir **[RETURN]**

Sullo schermo verrà visualizzata la DIRettrice di CP/M e le utilità COMandi, una delle quali è DISCKIT3. Si scriva:

disckit **[RETURN]**

Dopo pochi secondi si vedrà, in alto nello schermo, il messaggio iniziale di tale programma e successivamente il seguente:

One drive found

Questo messaggio indica che il programma di utilità DISC KIT è in esecuzione e che il computer ha scoperto che l'utente sta lavorando con un solo drive (quello all'interno del computer).
Se si è invece connesso anche il secondo drive il messaggio sarà:

Two drives found

In alto nello schermo si vedrà la seguente schermata

Copy	7	
Format	4	
Verify	1	
Exit from program	Ø	

Questo è il menù principale di DISC KIT. I numeri dei quadrati fanno riferimento ai tasti funzione situati sulla destra della tastiera (contrassegnati con **f0, f1, f4** e **f7**); scegliendo uno di tali tasti si seleziona la scelta corrispondente del menù.

Si noti che premendo a questo punto il tasto f0 si esce dal programma DISC KIT e si torna in ambiente CP/M (verrà visualizzato il prompt A>).

Vogliamo a questo punto formattare un disco, quindi si prema il tasto corrispondente alla funzione 4 (**f4**).

ATTENZIONE
LA FORMATTAZIONE DI UN DISCO REGISTRATO PRECEDENTEMENTE COMPORTA LA CANCELLAZIONE DI TUTTO IL SUO CONTENUTO

Si potrà vedere a questo punto sullo schermo un nuovo menù che permette di scegliere diversi formati:

System format		9
Data format		6
Vendor format		3
Exit menu		.

Come prima è possibile ora premere un tasto funzione (**f3**, **f6** o **f9**) per selezionare il tipo di formattazione desiderato. Ognuno di questi differenti tipi di formattazione verranno descritti successivamente nel manuale, per il momento si selezioni Data format premendo il tasto funzione 6.

Si noti che premendo il tasto . (sotto **f3**, **f6** e **f9**) si esce da tale menù per tornare nel menù principale di DISC KIT.

Avendo premuto il tasto funzione numero 6 (ed assumendo che non si sia connesso al computer alcun drive addizionale) si vedrà il messaggio:

A questo punto occorre togliere dal drive il disco di sistema CP/M ed inserire il disco che si desidera formattare. Il lato del disco che deve essere formattato deve avere l'etichetta rivolta verso l'alto.

Si prema ora il tasto **Y** (Y per Sì significa 'vai avanti a formattare il disco').

Il disco verrà formattato dalla traccia 0 alla 39; il numero di traccia attuale verrà visualizzato nell'angolo in alto a sinistra dello schermo.

Non sarà possibile formattare un disco se questo possiede il foro di protezione dalla scrittura aperto nel qual caso verrà visualizzato il seguente messaggio:

Disc write-protected
Insert disc to format
R-etry or C-ancel

... e occorrerà scrivere **C** per Cancellare, si dovrà togliere il disco ed inserire il disco esatto da formattare il quale deve avere il foro di protezione dalla scrittura chiuso.

Ci si assicuri di non chiudere il foro di protezione dalla scrittura in un disco di cui si vogliono tenere i programmi e di non chiudere MAI il foro di protezione dalla scrittura del disco di sistema CP/M.

Al termine della formattazione verrà richiesto di togliere il disco formattato dal drive e di premere un qualsiasi tasto per continuare.

Dopo aver fatto ciò sarà possibile formattare un altro disco inserendolo nel drive e premendo ancora il tasto **Y**. Si può ripetere consecutivamente questa operazione fino a quando non si sono formattati tutti i dischi necessari.

Al termine della formattazione è possibile tornare nel menù principale di DISC KIT premendo un qualsiasi tasto (diverso da **Y**).

L'opzione Copy e Verify verranno spiegate successivamente; per il momento si riavvii il computer usando i tasti **[CONTROL][SHIFT][ESC]**.

Si mantengano sempre le copie originali del disco di sistema CP/M in un luogo sicuro: esse sono realmente la chiave del sistema. Successivamente in questo manuale verrà spiegato come effettuare 'copie di lavoro' di tali dischi per poter tenere gli originali sempre al sicuro.

Formattazione sul secondo drive.

Si seguano le istruzioni fornite precedentemente, si selezioni l'opzione Format nel menù principale di DISC KIT premendo il tasto funzione **f4** e successivamente Data format premendo **f6**.

A questo punto verrà visualizzato sullo schermo un terzo menù che fornisce la possibilità di scegliere su quale drive si vuole formattare:

Scegliendo l'opzione Format B: (tasto **f5**) si potrà lasciare il disco di sistema (Side 1) nel Drive A e porre il disco da formattare nel Drive B.

Dopo aver selezionato Format B: si può premere **Y** per formattare o un qualsiasi altro tasto per tornare nel menù principale di DISC KIT.

Se si è selezionato invece Format A: (tasto **f8**) OCCORRE ricordarsi di togliere il disco di sistema dal Drive A ed inserivi il disco da formattare.

Non dimenticare - NON RISCHIARE MAI DI SOVRASCRIVERE IL DISCO DI SISTEMA CP/M.

Ora che abbiamo formattato un disco possiamo iniziare ad usare i programmi di BASIC.

Come salvare un programma residente in memoria su disco

Avendo scritto un programma in memoria salviamolo su disco scrivendo:

save “nomefile” **[RETURN]**

Si noti che è obbligatorio dare un nome al programma.

Il nomefile su disco è composto da due parti (campi). La prima parte è obbligatoria e può contenere al massimo otto caratteri: è possibile usare lettere e numeri ma NON spazi o simboli di punteggiatura. Il primo campo contiene di solito il nome del programma.

Il secondo campo è facoltativo. E' possibile usare al massimo tre caratteri ma non spazi o simboli di punteggiatura. I due campi sono separati da un punto.

Se il secondo campo non viene specificato i sistema attribuisce automaticamente il tipo; inserisce .BAS per i file di BASIC e .BIN per i file binari (linguaggio macchina).

Si provi a scrivere, come esempio, un breve programma, si inserisca un disco formattato nel drive e si scriva:

save “esempio” **[RETURN]**

Dopo pochi secondi sullo schermo apparirà la scritta Ready ed il programma sarà stato salvato su disco (in caso contrario si controlli sullo schermo il messaggio apparso, si potrebbe aver inserito il disco nel drive sbagliato, dimenticato il foro di protezione dalla scrittura chiuso o scritto male il comando).

Catalogo

Dopo aver salvato il programma precedente si scriva:

cat **[RETURN]**

Sullo schermo apparirà il seguente messaggio:

Drive A: user 0

EXAMPLE.BAS 1K

177K free

Viene visualizzato il nome del file, la lunghezza di questo e l'ammontare di spazio libero sul disco.

Caricamento da disco

I programmi possono essere caricati ed eseguiti da disco usando i comandi:

load "nomefile" **[RETURN]**
run **[RETURN]**

... o eseguiti direttamente usando il comando:

run "nomefile" **[RETURN]**

Si noti che i programmi protetti possono essere solo eseguiti direttamente.

Come salvare un programma su cassetta

Dopo aver scritto un programma lo si può salvare su cassetta scrivendo:

SAVE "NOMEFILE" **[RETURN]**

Il <nomefile> può essere composto da 16 caratteri (incluso lo spazio). Se si cerca di inserire un nome più lungo, il 17esimo carattere e tutti quelli che lo seguono verranno ignorati.

Il computer visualizzerà il messaggio:

Press REC e PLAY then any key:

Si ricordi ciò che è stato detto in precedenza riguardo al tasto da premere; il nastro inizierà a girare e il computer salverà il programma con il nome specificato.

IMPORTANTE

Il computer non può scoprire se si sono usati o meno i tasti corretti per l'avviamento del registratore, se quindi si è premuto solo il tasto **[PLAY]** il nastro girerà ed il programma visualizzerà un messaggio che indica che il programma è stato salvato mentre invece ciò non è vero.

ATTENZIONE: Se si premono per sbaglio i tasti **[REC]** e **[PLAY]** mentre si voleva invece leggere o caricare un programma, il computer cancellerà qualsiasi programma presente sulla cassetta. Se non lo si interrompe premendo il tasto **[ESC]** il nastro continuerà a girare e verranno quindi cancellati tutti i programmi ed i dati presenti in esso poiché il computer non trova il programma che stava cercando. Se si vogliono evitare questi inconvenienti è buona abitudine eliminare la linguetta di protezione delle cassette.

File senza nome e CAT

Se si salva un file senza fornirgli un nome:

SAVE""

... il BASIC lo salverà come Unnamed file (File senza nome). Su cassette è possibile salvare più file con lo stesso nome (inclusi i file senza nome), a differenza di un disco che richiede che ad ogni file venga fornito un nome diverso.

L'utente perderà facilmente traccia dei programmi memorizzati se non fornisce loro dei nomi che ricordino il contenuto: e si consiglia di aggiungere anche un codice al nome per ricordare qual'è la versione più recente dei programmi e dei file di dati.

E' possibile CATalogare i contenuti di una cassetta inserendo in comando CAT e seguendo la seguente istruzione:

Press PLAY then any key:

BASIC elencherà il contenuto del nastro visualizzando tutti i nomifile in maiuscolo seguiti dal numero di blocchi ed un singolo carattere che indica il tipo di file :

$ un programma BASIC
% un programma BASIC protetto
* un file di testo ASCII
& un file binario.

Il simbolo Ok al termine della linea indica che il file è leggibile e può quindi essere caricato dal computer. L'esecuzione della funzione CAT non intacca il programma attualmente in memoria.

Supersafe e Speedload

Il 464 offre due velocità di salvataggio: *Supersafe* a 1000 baud (bit di dati per secondo) e *Speedload* a 2000 baud. La velocità Speedload quindi scrive e legge ad una velocità doppia rispetto a Supersafe sacrificando però il margine di sicurezza necessario per utilizzare le cassette a basso costo (e di scarsa qualità). In questo caso potrebbero inoltre sorgere problemi di allineamento delle testine.

Per i programmi salvati e caricati sempre dallo stesso registratore, Speedload dovrebbe essere affidabile, sempre che si usino nastri di ottima qualità. Speedload dovrebbe essere inoltre in grado di caricare il software di cassette commerciali senza errori sebbene AMSTRAD avvisa che tale software potrebbe essere registrato usando le opzioni di entrambe le velocità
Il computer si predispone automaticamente alla velocità con cui è stato registrato il nastro. Quando si salva un programma è necessario indicare al computer se si desidera usare Speedload: in caso contrario verrà automaticamente assunto Supersafe.

Per selezionare la velocità più alta per salvare programmi e dati, si controlli che il computer sia in grado di ricevere istruzioni (deve essere visualizzato il prompt Ready) e si scriva:

SPEED WRITE 1

per tornare nella velocità più lenta si può riavviare il computer (in tal caso i dati verranno persi) oppure scrivere (al prompt Ready):

SPEED WRITE 0

Considerazioni sulle cassette

Sebbene il registratore accetta tutti i tipi di cassetta anche le C90, è necessario usare solo le C12 (sei minuti per lato) o al massimo le C30. I programmi memorizzati alla fine di cassette molto lunghe sono difficili da localizzare a meno che non ci si prepari ad aspettare fino a quando il computer li trovi (e ci si ricordi il nome fornitogli) oppure non si tenga meticolosamente traccia dell'indice del contanastro. Se si vuole riscrivere su un altro programma memorizzato nella cassetta occorre localizzare attentamente il punto in cui inizia il programma da cancellare e stare attenti a non scrivere su programmi che si vogliono tenere.

Soprattutto si usi una cassetta per pochi programmi. Le cassette C12 sono economiche e se si dovessero rovinare si avranno meno tentazioni voler salvare una cassetta buona solo in parte.

Si ricorda infine che il software commerciale viene venduto in strette condizioni di copyright. Non si devono effettuare copie o duplicare software (anche se da dare solo ad amici) fornito su cassette se non entro i termini di condizione di vendita del software (alcuni programmi incoraggiano ad effettuare copie di riserva). Le leggi di copyright sono state riesaminate per contrastare tutte le forme di duplicazione di software non autorizzato, e benchè vi sono stati fino ad ora pochi processi la situazione cambierà sostanzialmente nei prossimi anni e potrebbe essere retroattivamente applicabile.

Tecniche di salvataggio avanzate

Sul 464/6128 vi sono quattro modi di salvare un file, oltre a quello già noto del BASIC:

save "nomefile" **[RETURN]**

... vi sono altre tre metodi alternativi, ovvero:

File ASCII

save "nomefile",a **[RETURN]**

Aggiungendo il suffisso a si indica al computer di salvare il programma di dati in forma di file di testo ASCII. Questo metodo si applica ai file creati da altri editor e ad altre applicazioni di programma; il loro uso verrà discusso successivamente.

File protetti

save "nomefile",p **[RETURN]**

Aggiungendo il suffisso p si indica al computer di proteggere i dati in modo che il programma non possa essere LISTato dopo essere stato caricato (LOAD) o eseguito (RUN) e fermare l'esecuzione con il tasto **[ESC]**.

I programmi salvati in tal modo possono essere solo eseguiti direttamente usando il comando:

run "nomefile" **[RETURN]**

... oppure ...

chain "nomefile" **[RETURN]**

Se si pensa di aver bisogno successivamente di editare o modificare il programma, sarà necessario effettuare una copia di questo in forma non protetta, ovvero senza suffisso ,p.

File binari

save "nomefile",b, <indirizzo iniziale> , <lunghezza in byte> [,<voce opzionale>]**[RETURN]**

Questa opzione permette di salvare in forma binaria un blocco di dati presenti in memoria RAM su disco. E' necessario indicare esattamente al computer dove inizia la parte di memoria che si vuole salvare, quanto è lunga e, se richiesto, l'indirizzo della memoria in cui far partire l'esecuzione del file.

Screen Dump

Questa caratteristica binaria permette di salvare i dati presenti sullo schermo direttamente su disco. Il contenuto dello schermo verrà visualizzato così come appare usando il comando:

save "scrndump",b,49152,16384 **[RETURN]**

... dove 49152 è l'indirizzo iniziale della memoria dello schermo e 16384 quallo finale.

Per richiamare lo schermo si scriva:

load "scrndump" **[RETURN]**

Ulteriori informazioni sull'uso del sistema nel trattamento di file tra dischi (e cassette) verranno fornite successivamente.

Come copiare programmi da un disco in un altro

Usando i comandi visti in questo paragrafo possiamo vedere come eseguire una operazione di copia caricando semplicemente dal disco originale (sorgente) il programma in memoria, togliendo il disco originale e salvando il programma sul disco destinazione.

Parte 8 Modalità, Colori e Grafica ...

Il Personal Computer a colori 464/6128 Amstrad è dotato di tre modalità di visualizzazione delle immagini sullo schermo: Mode 0, Mode 1 e Mode 2.

Quando il computer viene acceso esso si trova automaticamente in Mode 1.

Per comprendere la differenza tra queste modalità si accenda il computer e si prema il tasto 1. Lo si mantenga premuto fino a quando non si sono riempite due linee di 1. Se si conta ora il numero di 1 presenti su una linea si vedrà che questo è pari a 40. Ciò significa che in modalità 1 (mode 1) si hanno 40 colonne. Si prema **[RETURN]**: si otterrà come risposta un messaggio Syntax error (Errore di sintassi) ma non ci si preoccupi, è solo un modo veloce per porre il computer in attesa di istruzioni. Infatti esso visualizzerà il messaggio Ready.

Si scriva ora:

mode 0 **[RETURN]**

Si potrà notare che i caratteri scritti sullo schermo sono molto larghi. Si prema ancora il tasto 1 e lo si mantenga premuto fino a quando non si sono riempite due linee di 1. Se si conta ora il numero di 1 presenti su una linea si vedrà che questo è pari a 20. Ciò significa che in modalità 0 (mode 0) si hanno 20 colonne. Si prema ancora **[RETURN]**.

Si scriva ora:

mode 2 [RETURN]

Si vedrà che questa è la modalità minore, infatti se si scrive una riga piena di 1 se ne potranno contare 80. Ciò significa che in modalità 2 (mode 2) vi sono 80 colonne.

Quindi:

Mode 0 = 20 colonne
Mode 1 = 40 colonne
Mode 2 = 80 colonne

Si prema infine ancora una volta **[RETURN]**.

COLORI

Vi è una scelta di 27 colori. Su un monitor a fosfori verdi (MM12) i colori vengono visualizzati in sfumature di verde.

In Mode 0, dei 27 colori disponibili è possibile visualizzarne contemporaneamente sullo schermo 16.
In Mode 1, dei 27 colori disponibili è possibile visualizzarne contemporaneamente sullo schermo 4.
In Mode 2, dei 27 colori disponibili è possibile visualizzarne contemporaneamente sullo schermo 2.

E' possibile modificare il colore del bordo della pagina (BORDER), dell'area in cui appaiono i caratteri (PAPER) o del carattere stesso (PEN) in modo indipendente l'uno dall'altro.

Nella Tabella 1 sono elencati i 27 colori disponibili ognuno dei quali ha associato il numero di riferimento dell'inchiostro (INK).

Per comodità questa tabella è riportata inoltre nel pannello in alto a destra del computer.

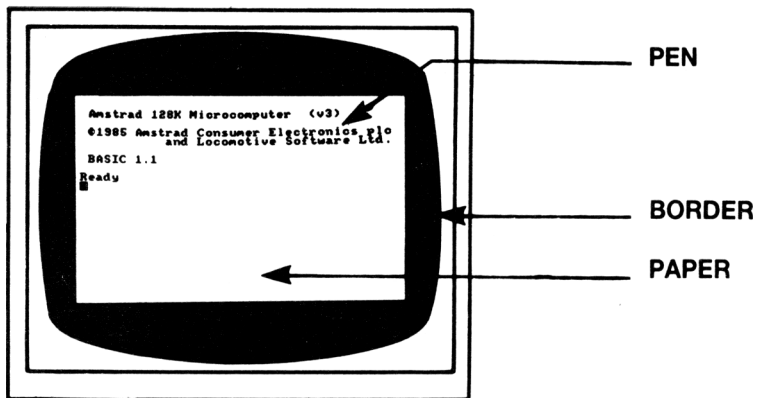
TABELLA DEI COLORI

N. inch.(INK)	Colore/Inch. (INK)	N. inch.(INK)	Colore/Inch. (INK)
0	Nero	14	Blu pastello
1	Blu	15	Arancio
2	Blu brillante	16	Rosa
3	Rosso	17	Magenta pastello
4	Magenta	18	Verde brillante
5	Malva	19	Verde mare
6	Rosso brillante	20	Azzurro brillante
7	Violetto	21	Verde limone
8	Magenta brillante	22	Verde pastello
9	Verde	23	Azzurro pastello
10	Azzurro	24	Giallo brillante
11	Blu cielo	25	Giallo pastello
12	Giallo	26	Bianco brillante
13	Bianco		

Tabella 1: numeri inchiostro e colori

Come appena spiegato quando si accende il computer questo è in Mode 1. Per riportarlo in tale modalità quando si è in un'altra si scriva:

mode 1 **[RETURN]**

Lo schermo di visualizzazione

BORDER è il bordo che racchiude l'area di visualizzazione dei caratteri (PAPER) (si noti che all'atto dell'accensione del computer sono entrambi blu). I caratteri visualizzati sullo schermo possono apparire solo all'interno di tale bordo. PAPER è l'area all'interno del bordo in cui vengono visualizzati i caratteri mentre PEN scrive i caratteri stessi.

Spiegheremo ora come vengono selezionati i colori che appaiono sullo schermo e come modificarli.

Quando si accende o si riavvia il computer il colore di BORDER è sempre equivalente al numero 1. Se si guarda nella Tabella 1 dei colori si potrà notare che tale numero equivale al blu. Il colore del bordo può essere modificato usando il comando BORDER a cui fa seguito un numero. Per cambiare il colore del bordo (BORDER) si scriva: border 13 **[RETURN]**

Fin qui tutto bene. Ora le cose si complicano leggermente ...

Quando si accende o si riavvia il computer vengono automaticamente selezionati i numeri 0 per PAPER e 1 per PEN. Ciò NON significa che si devono guardare i valori di 0 e 1 nella Tabella 1.

La cosa importante da ricordare è che sia 0 che 1 sono i numeri relativi a PAPER e a PEN: NON sono i numeri relativi al colore dell'inchiostro. Per capire tale differenza si provino ad immaginare 4 penne appoggiate sul proprio tavolo numerate 0,1,2 e 3 e si supponga di poterle riempire con uno qualsiasi dei 27 colori presenti nelle boccette dei colori numerate da 0 a 26. E'possibile inoltre che la penna numero 1 non sia necessariamente sempre dello stesso colore: questa può essere riempita con un inchiostro diverso, perciò è possibile riempire tutte e quattro le penne con lo stesso inchiostro.

Lo stesso discorso vale se applicato al computer. Usando i comandi PEN e INK è possibile selezionare il numero di penna e il colore per quella penna.

Si ricordi che stiamo operando in Mode 1 (40 colonne); guardando la Tabella 2 che segue si potrà vedere nella prima e terza colonna che il numero di PEN 1 corrisponde al colore numero 24. Se si guarda ora il numero di INK (inchiostro) 24 nella Tabella 1 si potrà vedere che tale colore è giallo brillante, ovvero il colore dei caratteri che appaiono sullo schermo quando si accende il computer.

PARAMETRI PER DIFETTO

Nr. schermo/carattere	**Colori Mode 0**	**Colori Mode 1**	**Colori Mode 2**
0	1	1	1
1	24	24	24
2	20	20	1
3	6	6	24
4	26	1	1
5	0	20	24
6	2	6	1
7	8	1	24
8	10	24	1
9	12	20	24
10	14	6	1
11	16	1	24
12	18	24	1
13	22	20	24
14	Lampeggiante 1,24	20	1
15	Lampeggiante 16,11	6	24

Tabella 2 : riferimenti PAPER/PEN/MODE/INK

Le relazioni PAPER/PEN/MODE/INK fornite nella Tabella 2 non sono comunque fisse. Vi sono dei parametri per difetto che vengono assunti ogni volta si accenda o si riavvii il computer. E' possibile modificarli usando il comando INK. Tale comando è composto da due parti (o 'parametri'). Il primo numero è il numero di PAPER o PEN a cui si vuole fornire un inchiostro, la seconda parte è l'inchiostro stesso. Le due parti del comando devono essere separate da una virgola ,

Ora che sappiamo che il numero di penna che stiamo usando è 1 modifichiamo il suo colore in arancione.

Si scriva:

ink 1,15 **[RETURN]**

... e si potrà vedere che i caratteri sullo schermo hanno cambiato colore.

Anche il colore dello sfondo può essere modificato usando un comando INK. Sappiamo che il numero di PAPER è 0: modifichiamone il colore in verde (colore numero 9) scrivendo il comando:

ink 0,9 **[RETURN]**

Ora usiamo un'altra penna. Si scriva:

pen 3 **[RETURN]**

Si noti che solo il colore dei nuovi caratteri (dopo il comando di modifica) è stato cambiato. Stiamo ora usando il numero di penna (**PEN**) 3. Guardando le Tabelle 1 e 2 si potrà notare che il numero 3 corrisponde al numero di colore 6 (rosso brillante). Modifichiamolo in rosa scrivendo:

ink 3,16 **[RETURN]**

Si ricordi che 3 è il colore della penna selezionata precedentemente con il comando: pen 3 e 16 è il colore dell'inchiostro, ovvero rosa.

Modifichiamo ora l'area in cui appaiono i caratteri. Quando si seleziona un nuovo PAPER il colore di sfondo precedente NON viene modificato poichè quel colore viene stampato da un PAPER differente. Per veder ciò si scriva:

paper 2 **[RETURN]**

Ancora una volta si usino le Tabelle 1 e 2 per vedere perchè il numero di fondo del numero PAPER 2 è azzurro. Lo si modifichi in nero scrivendo:

ink 2,0 **[RETURN]**

Sullo schermo ora si hanno caratteri scritti con numero di PEN 1 e 3 su sfondo con numero PAPER 0 e 2. Gli INK (inchiostri) che non si stanno usando possono essere modificati in PEN o PAPER. Si scriva, ad esempio:

ink 1,2 **[RETURN]**

... che modifica il colore di tutti i caratteri scritti precedentemente con PEN numero 1.

Si scriva:

cls [RETURN]

... per pulire lo schermo.

L'utente dovrebbe essere ora in grado di riportare il computer nella modalità originale (bordo e sfondo blu con caratteri giallo brillante) usando i comandi BORDER, PAPER, PEN e INK. Se così non fosse si riavvi il computer mantenendo premuti i tasti **[CONTROL][SHIFT]** e **[ESC]**.

COLORI LAMPEGGIANTI

E' possibile rendere il colore dei caratteri lampeggiante: che passa cioè da un colore ad un altro. Ciò è ottenibile mediante un ulteriore numero di colore inserito nel comando INK di PEN.

Per vedere lampeggiare i caratteri dello schermo dal colore bianco brillante al rosso brillante si riavvi il computer usando **[CONTROL][SHIFT][ESC]** e si scriva:

ink 1,26,6 **[RETURN]**

In questo caso, 1 è il colore del carattere (PEN) mentre 26 è il colore bianco brillante e 6 il colore alternativo, rosso brillante.

E' inoltre possibile rendere il colore dello schermo (PAPER) in fianco ai caratteri lampeggianti, lampeggiante anch'esso: da un colore in un altro. Ciò può essere ottenuto aggiungendo al comando INK dello schermo un ulteriore numero.

Per vedere lampeggiare lo schermo dal colore verde al giallo brillante si può scrivere:

ink 0,9,24 **[RETURN]**

In questo caso 0 è il numero di PAPER (schermo) mentre 9 è il colore verde e 24 il colore alternativo, giallo brillante.

Si riavvi a questo punto il computer mediante i tasti **[CONTROL][SHIFT][ESC].**

Si noti che in Mode 0 due dei caratteri (numeri 14 e 15) insieme a due dello schermo (numeri 14 e 15) sono i parametri per difetto dei colori lampeggianti. In altre parole non è necessario aggiungere un ulteriore numero al comando INK.

Si scriva:

mode 0 **[RETURN]**
pen 15 **[RETURN]**

si vedrà sullo schermo la parola Ready lampeggiare nei colori blu cielo e rosa.

Si scriva ora:

paper 14 **[RETURN]**
cls **[RETURN]**

Oltre alla parola Ready lampeggiante in blu cielo e rosa si potrà vedere che anche lo schermo lampeggia: modificando alternativamente il colore da giallo a blu.

I numeri 14 e 15 di PEN e di PAPER possono essere riprogrammati usando il comando INK in modo che si possano usare altri colori.

E' possibile, infine, rendere anche il bordo lampeggiante: per far ciò occorre aggiungere nel comando BORDER un ulteriore numero di colore. Si scriva:

border 6,9 **[RETURN]**

Si vedrà il bordo lampeggiare nei colori rosso brillante e verde. Si noti che il bordo può essere fissato in uno qualunque dei 27 colori senza dover tener conto della modalità in cui si sta operando: 0, 1 o 2.

Si riavvi a questo punto il computer mantenendo premuti i tasti **[CONTROL][SHIFT][ESC]**.

Per una ulteriore dimostrazione dei colori disponibili si scriva il seguente programma e lo si esegua (mediante comando run).

```
10 MODE 0 [RETURN]
20 num=600: REM fissa la velocità del programma [RETURN]
30 FOR b=0 TO 26 [RETURN]
40 LOCATE 3,12 [RETURN]
50 BORDER b [RETURN]
60 PRINT "colore del bordo";b [RETURN]
70FOR t=1 TO num [RETURN]
80 NEXT t,b [RETURN]
90 CLG [RETURN]
100 FOR p=0 TO 15 [RETURN]
110 PAPER p [RETURN]
120 PRINT "carta";p [RETURN]
130 FOR n=0 TO 15 [RETURN]
140 PEN n [RETURN]
150 PRINT "penna";n [RETURN]
160 NEXT n [RETURN]
170 FOR t=1 TO num*2 [RETURN]
180 NEXT t,p [RETURN]
190 MODE 1 [RETURN]
200 BORDER 1 [RETURN]
210 PAPER 0 [RETURN]
220 PEN 1 [RETURN]
230 INK 0,1 [RETURN]
240 INK 1,24 [RETURN]
run [RETURN]
```

IMPORTANTE

Nel programma precedente e in quelli che seguiranno nel manuale le parole chiave di BASIC sono state scritte in maiuscolo. Questo è anche il modo in cui appariranno sullo schermo quando verrà inserito un comando LIST. In generale è preferibile che l'utente scriva tali parole in minuscolo poichè nel caso di errori di battitura il listato del programma servirà ad evidenziarli; questo perchè la parole chiave di BASIC scritte in modo errato NON verranno convertite in maiuscolo.

Nel resto di questo capitolo scriveremo tali programmi sia in maiuscolo che in minuscolo per abituare l'utente a questo aspetto.

Il nome di una variabile come x o a$ NON verrà convertita in maiuscolo quando il programma viene listato sebbene il computer riconoscerà tale nome senza tener conto di come sia stata scritta: in maiuscolo o in minuscolo.

Attenzione

Da questo punto in poi, non verrà richiesto tutte le volte di premere **[RETURN]**; si assumerà infatti che questo venga fatto automaticamente dall'utente.

GRAFICA

Vi sono diversi simboli memorizzati in memoria: per visualizzarne uno si usa la parola chiave:

chr$().

Nelle parentesi tonde deve essere inserito un numero compreso tra 32 e 255.

Si riavvi il computer mantenendo premuti i tasti **[CONTROL][SHIFT][ESC]** e successivamente si scriva:

```
print chr$(250)
```

Non ci si dimentichi di premere **[RETURN]**. Verrà visualizzato sullo schermo il carattere corrispondente al numero 250 che è un uomo che cammina verso destra.

Per vedere sullo schermo tutti i caratteri associati ai numeri si scriva il seguente programma ricordandosi di premere **[RETURN]** al termine di ogni linea.

```
10 for n=32 to 255
20 print n;chr$(n);
30 next n
run
```

L'elenco dei caratteri corrispondenti ai numeri ed i loro simboli è riportato nel capitolo 'Riferimenti'.

LOCATE

Questo comando è usato per posizionare il cursore in un punto specifico dello schermo. A meno che non sia stato modificato da un comando locate, il cursore viene posizionato nell'angolo in alto a sinistra dello schermo che corrisponde alle coordinate x,y 1,1 (x è la posizione orizzontale e y la verticale). In Mode 1 vi sono 40 colonne e 25 linee. Per posizionare, in Mode 1, il cursore al centro della prima linea si devono usare le coordinate 20,1.

Si scriva (ricordando di premere **[RETURN]** al termine di ogni linea):

mode 1 .... lo schermo viene pulito, ed il cursore posizionato in alto a sinistra

```
10 locate 20,1
20 print chr$(250)
run
```

Per provare che il cursore si trova effettivamente sulla prima linea si scriva:

```
border 0
```

Il bordo sarà ora nero e si vedrà il simbolo dell'uomo in mezzo alla prima linea.

In Mode 0 vi sono solo 20 colonne ma 25 linee. Se si scrive ora:

```
mode 0
run
```

... si vedrà che il simbolo dell'uomo appare nell'angolo in alto a destra dello schermo. Questo perchè in modalità 0 la coordinata 20 corrisponde all'ultima colonna.

In Mode 2 vi sono 80 colonne e 25 linee. Usando lo stesso programma si potrà molto probabilmente immaginare dove apparirà ora l'uomo. Si scriva:

```
mode 2
run
```

Si ritorni ora in modo Mode 1 scrivendo:

```
mode 1
```

Si provino a questo punto a fare degli esperimenti modificando i comandi locate e chr$() e variando la posizione dei caratteri nello schermo. A titolo di esempio si provi a scrivere:

```
locate 20,12: print chr$(240)
```

Si vedrà una freccia in mezzo allo schermo. Si noti che in questa istruzione:

20 è la coordinata (x) orizzontale (nell'area 1- 40)
12 è la coordinata (y) verticale (nell'area 1 - 25)
240 è il numero corrispondente al simbolo (nell'area 32 - 255)

Per visualizzare il simbolo corrispondente al numero 250 in tutto lo schermo si scriva il seguente programma:

```
10 CLS
20 FOR x = 1 TO 39
30 LOCATE x,20
50 PRINT CHR$(250)
60 NEXT x
70 GOTO 10
run
```

Per fermare il programma si prema due volte il tasto **[ESC]**

Per eliminare il carattere precedente prima di visualizzarne un altro si scriva:

```
50 print " "; chr$(250)
```

(Tale linea sostituisce automaticamente la linea 50 precedentemente scritta).

Si scriva ora:

```
run
```

FRAME

Per migliorare il movimento del carattere nello schermo si aggiunga la seguente linea:

```
40 frame
```

Il comando FRAME sincronizza il movimento degli oggetti sullo schermo. Si ricordi che tale comando deve essere usato quando si vogliono far muovere i caratteri o le cifre sullo schermo in modo armonico.

Questo programma può essere ulteriormente migliorato aggiungendo alcuni ritardi ed usando diversi simboli.

Si scriva:

```
list
```

Si aggiungano ora le seguenti linee di programma:

```
70 FOR n = 1 TO 300 : NEXT n
80 FOR x = 39 TO 1 STEP-1
90 LOCATE x,20
100 FRAME
110 PRINT CHR$(251);" "
120 NEXT x
130 FOR n = 1 TO 300:NEXT n
140 GOTO 20
run
```

PLOT

Diversamente dal comando LOCATE, PLOT è usato per determinare la posizione del cursore grafico usando le coordinate pixel (abbreviazione di picture element). Un pixel è un segmento molto piccolo sullo schermo.

Si noti che il cursore grafico non è visibile ed è diverso dal cursore dei caratteri.

Vi sono 640 pixel orizzontali e 400 verticali. Le coordinate x,y sono posizionate a partire dall'angolo in basso a sinistra dello schermo: tale punto ha coordinate 0,0. Contrariamente al comando LOCATE le coordinate non differiscono nelle varie modalità (Mode 0, 1 o 2).

Per vedere ciò si riavvi il computer mantenendo premuti i tasti **[CONTROL][SHIFT][ESC]** e si scriva:

```
plot 320,200
```

In mezzo allo schermo verrà visualizzato un piccolo punto.

Si modifichi ora il Mode scrivendo:

```
mode 0
plot 320,200
```

Il punto sarà posizionato ancora in mezzo allo schermo ma è un po' più grande. Si modifichi ancora la modalità e si scriva lo stesso comando per vederne l'effetto in Mode 2. Si scriva:

```
mode 2
plot 320,200
```

Il punto è ancora in centro ma ancora più grande.

Per potersi rendere conto di persona dell'uso di tale comando si provi a posizionare diversi punti nello schermo nelle varie modalità. Al termine si ritorni in Mode 1 e si pulisca lo schermo scrivendo:

```
mode 1
```

DRAW

Si riavvi a questo punto il computer mantenendo premuti i tasti **[CONTROL][SHIFT][ESC]**. Il comando DRAW disegna una linea dalla posizione corrente del cursore grafico. Per vedere in dettaglio tale comando disegniamo un rettangolo sullo schermo usando il programma che segue.

Iniziamo a far ciò posizionando il cursore grafico mediante un comando PLOT. Disegnamo poi una linea partendo dalla posizione del cursore e verso l'angolo in alto a sinistra, poi da questo punto verso l'angolo destro, ecc.

Si scriva:

```
5 cls
10 plot 10,10
20 draw 10,390
30 draw 630,390
40 draw 630,10
50 draw 10,10
60 goto 60
run
```

Si prema due volte **[ESC]** per fermare il programma.

Si noti la linea 60 di tale programma; il computer indica di eseguire continuamente il programma partendo dalla linea 60: ciò viene fatto fino a quando non si preme due volte il tasto **[ESC]**. Questo comando è utile quando non si vuol far terminare automaticamente il programma e quindi visualizzare il prompt Ready sullo schermo.

Si aggiunga ora al programma le seguenti linee: si disegnerà all'interno del primo rettangolo un secondo. Si scriva:

```
60 plot 20,20
70 draw 20,380
80 draw 620,380
90 draw 620,20
100 draw 20,20
110 goto 110
run
```

Si prema due volte **[ESC]** per fermare il programma.

MOVE

Il comando MOVE opera in modo analogo al comando PLOT, il cursore grafico viene posizionato nelle coordinate x,y specificate; in questo caso comunque NON viene disegnato il punto.

Si scriva:

```
cls
move 639,399
```

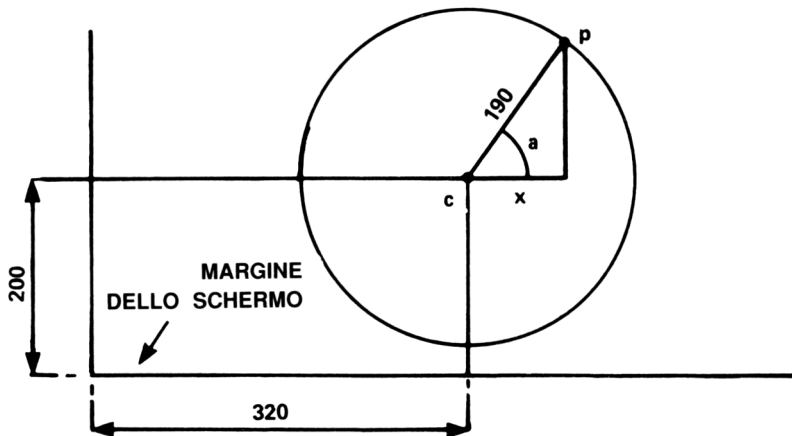
Sebbene non si veda alcun segno sullo schermo il cursore si è spostato nell'angolo in alto a destra.

Proviamo a disegnare una linea da quella posizione fino al centro dello schermo scrivendo:

```
draw 320,200
```

CERCHI

I cerchi possono essere sia posizionati sia disegnati. Un metodo per formare il cerchio consiste nel posizionare ogni punto x,y della circonferenza. Facendo riferimento alla figura sotto si può notare che il punto p della circonferenza può essere posizionato usando le coordinate x ed y. Queste sono:

```
x= 190*cos(a)
y = 190*sin(a)
```

Disegniamo i punti di un cerchio.

Nel programma precedente abbiamo disegnato i punti del cerchio rispetto all'angolo in basso a sinistra dello schermo. Se ora vogliamo posizionare il cerchio in mezzo allo schermo dobbiamo porre il centro del cerchio alle coordinate 320,320 e successivamente tutti i punti del cerchio rispetto a tale posizione.

Un tale programma potrebbe essere il seguente:

```
new
10 CLS
20 DEG
30 FOR a=1 TO 360
40 MOVE 320,200
50 DRAW 320+190*COS(a),200+190*SIN(a)
60 NEXT
run
```

Si noti l'uso della parola chiave NEW all'inizio del programma. Ciò indica al computer di pulire la memoria da qualsiasi programma (è un modo analogo a **[CONTROL][SHIFT][ESC]**). Tale comando, comunque, 'svuota' solo la memoria e non lo schermo.

Il raggio del cerchio può essere ridotto diminuendo il valore 190 (ovvero il numero di pixel).

Per vedere un diverso modo di posizionamento del cerchio (in radianti) si cancelli la linea 20 del programma scrivendo:

```
20
```

Per vedere disegnato un cerchio mediante linee che partono dal centro si editi la linea 50 e si sostituisca la parola plot con la parola draw. Tale linea dovrà essere come segue:

```
50 draw 320+190*cos(a), 200+190*sin(a)
```

Si provi ad apportare tale modifica e ad eliminare ancora la linea 20.

Si potrà notare che, nella linea 60 di tale programma, invece dell'istruzione NEXT a si è usato solo NEXT. Ciò è possibile in quanto in tale programma esiste una sola istruzione NEXT; il computer esegue tutte le istruzioni comprese tra FOR ed il NEXT ad esso associato. In programmi contenenti diversi cicli FOR NEXT si potrebbe voler aggiungere il nome della variabile dopo la parola NEXT al fine di identificare il corretto NEXT in fase di analisi del programma.

ORIGIN

Nel programma precedente abbiamo usato il comando MOVE per posizionare il centro di un cerchio e per aggiungervi le coordinate x,y. Invece di aggiungere tali coordinate al punto centrale disegnato è possibile usare il comando ORIGIN. Esso posizionerà il centro del cerchio e le coordinate di tutti i punti della circonferenza. Si provi a scrivere:

```
new
10 cls
20 for a = 1 to 360
30 origin 320,200
40 plot 190*cos(a),190*sin(a)
50 next
run
```

Per disegnare quattro piccoli cerchi sullo schermo si scriva il seguente programma:

```
new
10 CLS
20 FOR a = 1 TO 360
30 ORIGIN 196,282
40 PLOT 50*COS(a),50*SIN(a)
50 ORIGIN 442,282
60 PLOT 50*COS(a),50*SIN(a)
70 ORIGIN 196,116
80 PLOT 50*COS(a),50*SIN(a)
90 ORIGIN 442,116
100 PLOT 50*COS(a),50*SIN(a)
110 NEXT
run
```

Per vedere un diverso modo di creazione di cerchi si scriva il seguente programma:

```
new
10 MODE 1
20 ORIGIN 320,200
30 DEG
40 MOVE 0,190
50 FOR a=0 TO 360 STEP 10
60 DRAW 190*SIN(a),190*COS(a)
70 NEXT
run
```

Questa volta viene disegnata una linea di coordinata in coordinata lungo tutta la circonferenza del cerchio. Si noti come il cerchio viene disegnato più velocemente.

Si osservi ancora una volta l'effetto che si ottiene eliminando il comando DEG, cancellando la linea 30 ed eseguendo di nuovo il programma.

FILL

Il comando FILL viene usato per riempire un'area dello schermo racchiusa da un disegno o da un'angolo dello schermo grafico.

Si riavvi il computer mediante i tasti **[CONTROL][SHIFT][ESC]** e si scriva:

```
new
10 cls
20 move 20,20
30 draw 620,20
40 draw 310,380
50 draw 20,20
run
```

Sullo schermo sarà possibile vedere un triangolo. Si sposti il cursore grafico in centro allo schermo scrivendo:

```
move 320,200
```

Usando la parola chiave FILL seguita dal numero di penna, ad esempio 3, riempiremo ora lo schermo usando la penna specificata e iniziando dalla posizione del cursore (centro dello schermo) fino al confine. Si scriva:

```
fill 3
```

Si sposti ora il cursore grafico fuori dal triangolo scrivendo:

```
move 0,0
```

Si guardi ora cosa accade scrivendo:

```
fill 2
```

Il computer ha usato la penna numero 2 per riempire l'area compresa tra le linee disegnate e gli angoli dello schermo.

Modifichiamo ora il programma scrivendo le seguenti linee:

```
50 draw 50,50
60 move 320,200
70 fill 3
run
```

Si noterà che ogni apertura nei confini permetterà all'inchiostro di filtrare!

Questo punto è dimostrato ulteriormente riempiendo prima un cerchio e poi disegnandone un altro. Si scriva:

```
new
10 CLS
20 FOR a=1 TO 360
30 ORIGIN 320,200
40 PLOT 190*COS(a),190*SIN(a)
50 NEXT
60 MOVE -188,0
70 FILL 3
run
```

Si scriva ora:

```
new
10 MODE 1
20 ORIGIN 320,200
30 DEG
40 MOVE 0,190
50 FOR d=0 TO 360 STEP 10
60 DRAW 190*SIN(d),190*COS(d)
70 NEXT
80 MOVE -188,0
90 FILL 3
run
```

Possiamo rendere il profilo del cerchio invisibile ponendo l'inchiostro della penna dello stesso colore dell'inchiostro della carta. Aggiungiamo:

```
45 GRAPHICS PEN 2:INK 2,1
run
```

Il comando GRAPHICS PEN seleziona la penna da usare per disegnare il grafico sullo schermo. Il comando INK specifica il colore per tale penna che è, in questo caso, uguale a quella della penna (cioè il colore numero 1).

Si scriva infine il seguente programma dimostrativo:

```
new
10 MODE 0:BORDER 13
20 MOVE 0,200:DRAW 640,200
30 FOR x=80 TO 560 STEP 80
40 MOVE x,0:DRAW x,400
50 NEXT:MOVE -40,300
60 FOR c=0 TO 7
70 MOVER 80,0: FILL c
80 MOVER 0,-200:FILL c+8
90 MOVER 0,200:NEXT
100 GOTO 100
run
```

I colori dell'area riempita possono essere modificati scrivendo:

```
100 SPEED INK 30,30
110 BORDER RND*26,RND*26
120 INK RND*15,RND*26,RND*26
130 FOR t=1 TO 500:NEXT:GOTO 110
run
```

Ulteriori dettagli...

Per ulteriori informazioni sulla grafica del 464/6128 si consulti la parte "Parliamo della grafica" nel capitolo "A vostra disposizione".

Per concludere questo paragrafo seguono alcuni programmi dimostrativi che includono molti comandi e parole chiave la cui conoscenza è molto utile. Tali programmi disegnano continuamente oggetti sullo schermo.

```
new
10 BORDER 0:GRAPHICS PEN 1
20 m=CINT(RND*2):MODE m
30 i1=RND*26:i2=RND*26
40 IF ABS(i1-i2)<10 THEN 30
50 INK 0,i1:INK 1,i2
60 s=RND*5+3:ORIGIN 320,-100
70 FOR x=-1000 TO 0 STEP s
80 MOVE 0,0:DRAW x,300:DRAW 0,600
90 MOVE 0,0:DRAW -x,300:DRAW 0,600
100 NEXT:FOR t=1 TO 2000:NEXT:GOTO 20
run
```

```
10 MODE 1:BORDER 0:PAPER 0
20 GRAPHICS PEN 2: INK 0,0:i=14
30 EVERY 2200 GOSUB 150
40 indice=0:CLG
50 INK 2,14+RND*12
60 b%=RND*5+1
70 c%=RND*5+1
80 ORIGIN 320,200
90 FOR a=0 TO 1000 STEP PI/30
100 x%=100*COS(a)
110 MOVE x%,y%
120 DRAW 200*COS(a/b%),200*SIN(a/c%)
130 IF indice=1 THEN 40
140 NEXT
150 indice=1:RETURN
run
```

```
10 MODE 1:BORDER 0:DEG
20 PRINT "Attendere per favore"
30 FOR n=1 TO 3
40 INK 0,0:INK 1,26:INK 2,6:INK 3,18
50 IF n=1 THEN sa=120
60 IF n=2 THEN sa=135
70 IF n=3 THEN sa=150
80 IF n=1 THEN ORIGIN 0,-50,0,640,0,400 ELSE ORIGIN 0,0,0,640,0,400
90 DIM cx(5),cy(5),r(5),lc(5)
100 DIM np(5)
110 DIM px%(5,81),py%(5,81)
120 st=1:cx(1)=320:cy(1)=200:r(1)=80
130 FOR st=1 TO 4
140 r(st+1)=r(st)/2
150 NEXT st
160 FOR st=1 TO 5
170 lc(st)=0:np(st)=0
180 np(st)=np(st)+1
190 px%(st,np(st))=r(st)*SIN(lc(st))
200 py%(st,np(st))=r(st)*COS(lc(st))
210 lc(st)=lc(st)+360/r(st)
220 IF lc(st) < 360 THEN 180
230 px%(st,np(st)+1)=px%(st,1)
240 py%(st,np(st)+1)=py%(st,1)
250 NEXT st
260 CLS:cj=REMAIN(1):cj=REMAIN(2)
270 cj=REMAIN(3):INK 1,2:st=1
280 GOSUB 350
290 LOCATE 1,1
300 EVERY 25,1 GOSUB 510
310 EVERY 15,2 GOSUB 550
320 EVERY 5,3 GOSUB 590
330 EVERY cx,cy,r,lc,no,px%,py%:NEXT
340 GOTO 340
350 cx%=cx(st):cy%=cy(st):lc(st)=0
360 FOR x%=1 TO np(st)
370 MOVE cx%,cy%
380 DRAW cx%+px%(st,x%),cy%+py%(st,x%),1+(st MOD 3)
390 DRAW cx%+px%(st,x%+1),cy%+py%(st,x%+1),1+(st MOD 3)
400 NEXT x%
410 IF st=5 THEN RETURN
420 lc(st)=0
430 cx(st+1)=cx(st)+1.5*r(st)*SIN(sa+lc(st))
```

Questo programma continua nella prossima pagina.

```
440 cy(st+1)=cy(st)+1.5*r(st)*COS(sa+lc(st)
450 st=st+1
460 GOSUB 350
470 st=st-1
480 lc(st)=lc(st)+2*sa
490 IF (lc(st) MOD 360)<>0 THEN 430
500 RETURN
510 ik(1)=1+RND*25
520 IF ik(1)=ik(2) OR ik(1)=ik(3) THEN 510
530 INK 1,ik(1)
540 RETURN
550 ik(2)=1+RND*25
560 IF ik(2)=ik(1) OR ik(2)=ik(3) THEN 550
570 INK 2,ik(2)
580 RETURN
590 ik(3)=1+RND*25
600 IF ik(3)=ik(1) OR ik(3)=ik(2) THEN 590
610 INK 3,ik(3)
620 RETURN
```

Parte 9: Uso del Suono...

Gli effetti sonori sono generati da una coppia di altoparlanti contenuti nel monitor.

Il livello del suono può essere regolato per mezzo del controllo di **VOLUME** posto sul retro del computer. E' possibile inoltre collegare l'uscita STEREO del computer a quella del proprio stereo. Ciò permetterà di sentire il suono emesso dal computer attraverso le casse acustiche del proprio stereo o mediante cuffia. Ulteriori informazioni sulla connessione di tale presa STEREO verranno fornite nella parte 2 di questo Corso introduttivo.

Il comando SOUND

Il comando SOUND ha sette parametri. I primi due devono essere specificati; gli altri sono opzionali. Il comando deve essere scritto come:

SOUND <stato del canale>, <periodo della nota>, <durata>, <volume>, <inviluppo del volume>, <inviluppo del tono>, <periodo di rumore>.

Sembra complicato, ma se si analizza ogni parametro, si potrà notare che non è così. Guardiamo ad uno ad uno tutti i parametri...

Stato del canale

Per mantenere le cose più semplici possibili si guardi allo stato del canale come ad un numero del canale sonoro. Vi sono tre canali; per il momento useremo lo stato di canale numero 1.

Periodo della nota

Il periodo della nota è un modo tecnico per definire la tonalità del suono o, in altre parole, "la nota" (cioè do re mi fa, ecc.). Ogni nota ha associato un numero e tale numero è il periodo della nota. Facendo riferimento al capitolo 'Riferimenti' si vedrà che la nota DO centrale ha un periodo di 239.

Si riavvi il computer premendo i tasti **[CONTROL][SHIFT][ESC]** e si scriva:

```
10 sound 1,239
run
```

Si udirà una breve nota corrispondente al DO centrale della durata di 0,2 secondi.

Se non si sente alcun suono ci si assicuri che il controllo di **VOLUME** del computer non sia a zero. Si riscriva il programma precedente e lo si esegua (run).

Durata

Questo parametro fissa la lunghezza del suono, in altre parole 'la durata'. Il parametro opera in unità di 0.01 (un centesimo) secondi; se non si specifica la durata questa sarà pari a 20: questa è la ragione per cui le note udite durano 0.2 secondi, cioè 0.01 moltiplicato per 20.

Per far durare una nota 1 secondo si deve usare 100; per 2 secondi si deve usare 200. Si scriva:

```
10 sound 1,239,200
run
```

Si udirà la nota Do centrale per la durata di 2 secondi.

Volume

Questo parametro specifica il volume iniziale della nota. Il numero può essere compreso tra 0 e 15. Il volume 0 è il minimo, mentre 15 è il massimo. Se non si specifica alcun numero viene assunto per difetto 12. Si scriva:

```
10 sound 1,239,200,5
run
```

Si noti il volume di tale suono. Lo si riscriva ora aumentano il volume:

```
10 sound 1,239,200,15
run
```

Si potrà notare che tale volume è molto più alto.

Inviluppo del volume

Al fine di far variare il volume all'interno della durata della nota è possibile specificare un inviluppo del volume usando un diverso comando: ENV. E' possibile infatti creare diversi inviluppi di volume e, come nel comando SOUND, ogni inviluppo ha un numero di riferimento. Se si è creato un inviluppo di volume con numero di riferimento 1 e lo si desidera usare nel comando SOUND, quando viene richiesto il parametro "inviluppo del volume" si scrive 1. Tra breve verrà fornita la spiegazione riguardante la creazione dell'inviluppo del volume.

Inviluppo del tono

Al fine di far variare il tono all'interno della durata della nota è possibile specificare un inviluppo del tono usando un diverso comando: ENT. E' possibile infatti creare diversi inviluppi di tono e, come nel comando SOUND, ogni inviluppo ha un numero di riferimento. Se si è creato un inviluppo di tono con numero di riferimento 1 e lo si desidera usare nel comando SOUND, quando viene richiesto il parametro "inviluppo del tono" si scrive 1. Tra breve verrà fornita la spiegazione riguardante la creazione dell'inviluppo del tono.

Rumore

"Rumore" è l'ultimo parametro del comando SOUND. La gamma di rumori disponibili è compresa tra 1 e 31. Si provi ad aggiungere alla fine del comando sound il numero di rumore 2 e si sentano gli effetti. Si modifichi il numero di "rumore" ponendolo a 27 e si senta la differenza. Si scriva:

```
10 sound 1,239,200,15,,,2
```

Si notino i due parametri 'nulli' (,,,) prima del parametro 2. Questo è dovuto al fatto che non si è creato un <inviluppo di volume> e nemmeno un <inviluppo di tono>.

Creazione di un inviluppo di volume.

Il comando di inviluppo del volume è ENV. Nella forma più semplice tale comando ha 4 parametri. Il comando viene scritto come:

ENV <numero di inviluppo>, <numero di passi>, <ampiezza del passo>, <tempo del passo>.

Analizziamone uno alla volta.

Numero dell'inviluppo

E' un numero di riferimento (compreso tra 0 e 15) dato ad un particolare inviluppo di volume richiamabile in un comando SOUND.

Numero di passi

Questo parametro specifica i differenti passi di volume prima del termine del suono. In una nota, ad esempio, della durata di 10 secondi, si potrebbero volere 10 passi di volume di 1 secondo ciascuno. In tal caso, il parametro <numero di passi> deve essere 10.

La gamma di tali numeri è compresa tra 0 e 127.

Ampiezza dei passi

Ogni passo può variare in ampiezza da un livello di volume compreso tra 0 e 15 tenendo conto dell'ultimo passo. I 16 differenti livelli di volume sono uguali a quelli del comando SOUND. La gamma di ampiezza di tali numeri varia da -128 a +127; il livello di volume ritorna a zero ogni volta raggiunge valore 15.

Tempo del passo

Questo numero specifica l'unità di tempo intercorrente tra due passi in 0,001 secondi (1/100esimo di secondo). La gamma di tale passo varia da 0 a 256 ciò significa che il tempo più lungo tra i passi è quindi di 2.56 secondi (0 viene trattato come 256).

Si noti comunque, che il parametro <numero di passi> moltiplicato per il <tempo del passo>, in un comando SOUND, non deve essere maggiore della <durata>; in caso contrario, il suono terminerà prima che siano stati effettuati tutti i passi di volume (in tal caso i passi di volume eccedenti verranno scaricati).

Inoltre se, in un comando SOUND, il parametro <durata> è maggiore del <numero di passi> moltiplicato per il <tempo del passo> il suono continuerà anche quando sono stati effettuati tutti i passi di volume e rimarrà constantemente sul livello finale.

A titolo di esperimento, si provi a scrivere il seguente programma:

```
10 env 1,10,1,100
20 sound 1,142,1000,1,1
run
```

La linea 20 specifica un suono con periodo della nota di 142, di durata 10 secondi con volume iniziale di 1, un inviluppo di volume pari a 1, composto da 10 passi, con aumento di volume ad ogni passo 1, ogni secondo (100x0,01 secondi).

Si modifichi la linea 10 in ognuno dei seguenti modi e si esegua ogni volta il programma per sentire l'effetto di modifica dell'inviluppo.

```
10 env 1,100,1,10
10 env 1,100,2,10
10 env 1,100,4,10
10 env 1,50,20,20
10 env 1,50,2,20
10 env 1,50,15,30
```

Ed infine si provi la seguente:

```
10 env 1,50,2,10
```

Si sarà notato che dimezzando il suono il livello resta costante. Questo perchè il numero di passi era 50 ed il tempo per ogni passo pari a 0,1 secondo. Del resto, la lunghezza dell'ampiezza variava solo di 5 secondi mentre la durata del suono nel comando SOUND nella linea 20 era pari a 1000 cioè 10 secondi.

Si provi ad effettuare delle prove per vedere il tipo di suono che si riesce a creare.

Se si desidera creare un inviluppo di volume più complicato i 3 parametri: <numero di passi>, <ampiezza del passo> e <tempo del passo> possono essere ripetuti alla fine di ogni comando ENV fino a quattro volte, per specificare una differente 'sezione' dello stesso inviluppo.

Creazione di un inviluppo del tono

Il comando di inviluppo del tono è ENT. Nella forma più semplice, il comando ha quattro parametri. Il comando viene scritto come:

ENT <numero di inviluppo>, <numero di passi>, <ampiezza della nota>, <tempo del passo>.

Numero dell'inviluppo

Questo è un numero di riferimento (tra 0 e 15) di un particolare inviluppo: può essere quindi specificato nel comando SOUND.

Numero di passi

Questo parametro specifica quanti passi di tono devono essere eseguiti prima che termini il suono. In una nota, ad esempio, della durata di dieci secondi, si potrebbero volere dieci passi di tono; cioè uno ogni secondo. In questo caso il parametro <numero di passi> deve essere uguale a dieci.

La gamma di tali numeri è compresa tra 0 e 239.

Ampiezza della nota

Ogni passo di tono può variare tra -128 e +127 tenendo conto dell'ultimo passo. I passi negativi incrementano la frequenza della nota; quelli positivi la decrementano. L'ampiezza minore equivale a 0. Questo deve essere ricordato quando si calcola l'inviluppo di tono. La gamma completa del periodo del tono è riportata nel capitolo 'Riferimenti'.

Tempo del passo

Questo numero specifica l'unità di tempo intercorrente tra due passi in 0,001 secondi (1/100esimo di secondo). La gamma di tale passo varia da 0 a 255 ciò significa che il tempo più lungo tra i passi è quindi di 2.56 secondi (0 viene trattato come 256).

Si noti comunque, che il parametro <numero di passi> moltiplicato per il <tempo del passo>, in un comando SOUND, non deve essere maggiore della <durata>, in caso contrario, il suono terminerà prima che siano stati effettuati tutti i passi di tono (in tal caso i passi di tono eccedenti verranno scaricati).

Inoltre, se in un comando SOUND, il parametro <durata> è maggiore del <numero di passi> moltiplicato per il <tempo del passo> il suono continuerà anche quando sono stati effettuati tutti i passi di tono e rimarrà constantemente sul livello finale.

A titolo di esperimento, si provi a scrivere il seguente programma:

```
10 ent 1,100,2,2
20 sound 1,142,200,15,,1
run
```

La linea 20 specifica un suono con periodo della nota di 142, di durata 2 secondi con volume iniziale di 15 (massimo), senza inviluppo di volume (rappresentato dal parametro nullo ,,) e con inviluppo della nota pari a 1.

La linea 10 ha numero di inviluppo della nota uguale a 1 formato da 100 passi, con incremento del periodo della nota (riduzione della frequenza) di 2 ogni 0,02 secondi (2x0.01 secondi).

Si modifichi la linea 10 in ognuno dei seguenti modi e si esegua ogni volta il programma per sentire l'effetto di modifica dell'inviluppo.

```
10 ent 1,100,-2,2
10 ent 1,10,4,20
10 ent 1,10,-4,20
```

Ora si sostituisca il comando sound e l'inviluppo della nota scrivendo:

```
10 ent 1,2,17,70
20 sound 1,71,140,15,,1
30 goto 10
run
```

Si prema il tasto **[ESC]** due volte per fermare il programma.

Ora è possibile porre in un comando sound l'inviluppo del volume, e l'inviluppo della nota per creare diversi suoni. Si scriva:

```
new
10 env 1,100,1,3
20 ent 1,100,5,3
30 sound 1,142,300,1,1,1
run
```

Si sostituisca a questo punto la linea 20 scrivendo:

```
20 ent 1,100,-2,3
run
```

Si sostituiscano ora tutte le linee scrivendo:

```
10 env 1,100,2,2
20 ent 1,100,-2,2
30 sound 1,142,200,1,1,1
run
```

Se si desidera creare un inviluppo di tono più complicato i 3 parametri: <numero di passi>, <ampiezza del passo> e <tempo del passo> possono essere ripetuti alla fine di ogni comando ENT fino a quattro volte, per specificare una differente 'sezione' dello stesso inviluppo.

Si provino ad effettuare delle variazioni. Si provi ad aggiungere, nel comando SOUND, del <rumore> ed alcuni parametri all'inviluppo del volume e a quello del tono.

Il Capitolo 3 contiene tutti i dettagli sui vari comandi relativi al suono. Se si è interessati a questo aspetto si veda il paragrafo 'Il sound della musica' nel Capitolo 8.

Parte 10: Introduzione ad AMSDOS e CP/M (solo 6128)...

Cos'è AMSDOS?

Quando un computer viene acceso o riavviato, il sistema esegue automaticamente 'AMSDOS'. AMSDOS è l'abbreviazione di AMstrad Disc Operation System (Disco di sistema operativo AMSTRAD) e permette di usare i seguenti comandi e funzioni per il trattamento dei file:

LOAD "nome file"
RUN "nome file"
SAVE "nome file"
CHAIN "nome file"
MERGE "nome file"
CHAIN MERGE "nome file"
OPENIN "nome file"
OPENOUT "nome file"
CLOSEIN
CLOSEOUT
CAT
EOF
INPUT #9
LINE INPUT #9
LIST #9
PRINT #9
WRITE #9

Copia di un intero disco

E' possibile copiare il contenuto di un intero disco su un altro usando il programma **DISCKIT** del lato 1 del disco di sistema CP/M.

E' possibile usare tale metodo per effettuare copie di riserva dei dischi di sistema originali stessi.

Si inserisca il lato 1 del disco di sistema nel drive e si scriva:

|cpm

Dopo il prompt A> si scriva:

disckit

Dopo pochi secondi si vedrà in alto sullo schermo il messaggio iniziale di DISC KIT seguito immediatamente dal seguente:

One drive found

(Ovvero: Trovato un drive).

Questo messaggio indica che il computer sta eseguendo il programma di utilità DISC KIT e che ha scoperto che si sta operando con un solo drive (quello all'interno del computer). Se si è connesso un drive addizionale il messaggio sarà:

Two drive found

(Ovvero: Trovati due drive)

In fondo allo schermo si potrà vedere ciò che segue:

Copy	7	
Format	4	
Verify	1	
Exit from program	Ø	

Questo è il menù principale di DISC KIT. I numeri dei quadrati fanno riferimento ai tasti funzione situati sulla destra della tastiera (contrassegnati con **f0**, **f1**, **f4** e **f7**); scegliendo uno di tali tasti si seleziona la scelta corrispondente del menù.

Si noti che premendo a questo punto il tasto f0 si esce dal programma DISC KIT e si torna in ambiente CP/M (verrà visualizzato il prompt A>).

Vogliamo a questo punto copiare un disco, si prema quindi il tasto corrispondente alla funzione 7 (**f7**).

ATTENZIONE

LA COPIA IN UN DISCO REGISTRATO PRECEDENTEMENTE COMPORTA LA CANCELLAZIONE DI TUTTO IL SUO CONTENUTO

Copia di un sistema ad un solo drive

Assumendo che si stia usando un sistema ad un solo drive (cioè NON si sia connesso un ulteriore drive), si vedrà sullo schermo il seguente messaggio:

(ovvero Y Copia
qualsiasi tasto per uscire dal menù)

A questo punto occorre togliere il disco originale CP/M ed inserire il disco del quale si vuole effettuare la copia. Se si desidera copiare il disco di sistema CP/M lo si lasci all'interno del drive.

Quando il disco di cui si vuole effettuare la copia è nel drive, si prema il tasto **Y** (Y per Yes ovvero Sì indica di proseguire l'operazione di copia).

Verrà esaminato il formato del disco e visualizzato, a titolo informativo, sullo schermo in alto.

Subito dopo verrà visualizzato il seguente messaggio:

Insert disc to Write	(ovvero: Inserire un disco da Scrivere
Press any key to continue	Si prema un qualunque tasto per continuare)

A questo punto occorre togliere dal drive il disco di cui si vuole fare la copia ed inserire il disco su cui si desidera copiare le informazioni e premere un qualsiasi tasto.

Verrà esaminato il formato del disco e visualizzato (anche se non è formattato), a titolo informativo, sullo schermo in alto.

Se il disco su cui si vuole copiare l'informazione non è formattato correttamente (o non lo è del tutto) in fase di copia verrà visualizzato il seguente messaggio:

Disc isn't formatted (or faulty)	(ovvero: Disco non formattato (o difettoso)
Going to format while copying	Formattazione in fase di copia
Disc will be system format	Il disco verrà formattato in formato sistema)

.. o uno simile a seconda dei dischi che si stanno copiando.

Quando il computer è di nuovo pronto per leggere altre informazioni dal disco originale visualizzerà il seguente messaggio:

Insert disc to READ	(ovvero: Inserire il disco da leggere
Press any key to continue	Si prema un qualunque tasto per continuare)

... ed occorre reinserire il disco che si vuole copiare.

Dopo aver ripetuto questo processo diverse volte, il contenuto del primo disco sarà stato copiato nel secondo, ed il messaggio:

Copy completed	(Ovvero: copia completata
Remove disc	togliere il disco
Press any key to continue	premere un qualunque tasto per continuare)

... verrà visualizzato sullo schermo dopo del quale si devono seguire le istruzioni sullo schermo, che danno la possibilità di copiare un altro disco (premendo Y) o 'exit' (uscire) dal menù principale di DISC KIT.

Protezione da scrittura

Non sarà possibile copiare informazioni su un disco se questo possiede il foro di protezione dalla scrittura aperto, nel qual caso verrà visualizzato il seguente messaggio:

Disc write-protected
Insert disc to WRITE
R-etry or C-ancel

(ovvero: Disco protetto da scrittura
Inserire un disco da Scrivere
R-iprova o C-ancella)

... e occorrerà scrivere C per Cancellare, si dovrà togliere il disco ed inserire il disco esatto su cui copiare, il quale deve avere il foro di protezione dalla scrittura chiuso.

Ci si assicuri di non chiudere il foro di protezione dalla scrittura in un disco di cui si vogliono tenere i programmi e di non chiudere MAI il foro di protezione dalla scrittura del disco di sistema CP/M.

Verifica dei dischi

Il programma DISCKIT permette inoltre di verificare (controllare) un disco.
Vengono visualizzate sullo schermo tutte le informazioni del disco e vengono letti tutti i file del disco.

Qualsiasi errore presente nei file viene riportato sullo schermo.

Per verificare un disco si inserisca il lato 1 del disco di sistema e si scriva:

|cpm
Al prompt A> si scrive:

disckit

Si seleziona l'opzione Verify del menù principale di DISC KIT (tasto **f1**) e si seguono le istruzioni presenti sullo schermo.

Se si possiede un sistema a due drive viene fornita la possibilità di scegliere il Drive A o il Drive B.

Quando il disco che si vuole controllare è stato inserito nel drive si preme Y.

Quando la verifica è stata completata sullo schermo appare il seguente messaggio:

Verify completed	(Ovvero: Verifica completata
Remove disc	Togliere il disco
Press any key to continue	Premere un qualsiasi tasto per continuare).

Se si possiede un sistema a doppio drive verrà richiesto di estrarre ENTRAMBI i dischi prima di eseguire ulteriori operazioni. Si può ripetere il processo di verifica premendo il tasto Y oppure si può uscire dal menù principale di DISC KIT premendo un qualsiasi tasto.

Parte 11: Introduzione al Bank Manager (solo 6128)...

Uso del secondo gruppo di 64k di memoria

Il 6128 contiene 128K di RAM (Random Access Memory) suddivisa in due lotti di 64k. CP/M Plus usa sempre 128K mentre BASIC non usa normalmente il secondo lotto di 64K: la sola memoria disponibile è quindi il primo lotto di 64K. Sarebbe davvero un peccato lasciare questi 64K inutilizzati in fase di esecuzione di BASIC: per questo motivo viene fornito un particolare programma che permette un uso speciale di tale memoria. Il programma mette a disposizione alcuni comandi che rendono possibile l'uso del secondo lotto di 64K di RAM così come uno spazio di memoria per le immagini dello schermo o come uno spazio di memoria per le stringhe.

Il programma che mette a disposizione questi comandi si chiama "BANK MANAGER" dove "bank" è un termine tecnico usato per descrivere una parte di memoria.

Uso di BANK MANAGER per la gestione delle immagini.

Il 6128 visualizza uno schermo alla volta. Per far ciò necessita di 16K di memoria nel quale memorizza le informazioni riguardo il colore e la luminosità di ogni pixel (punto) sullo schermo. La memoria del 6128 permette di memorizzare un massimo di sei immagini contemporaneamente (ogni immagine è memorizzata su 16K). BANK MANAGER permette di impostare e visualizzare almeno cinque dei sei schermi possibili di BASIC.

Quando si accende il computer, lo schermo visualizzato proviene da un blocco di 16K di memoria (chiamato "Blocco 1") del primo lotto di 64K. Gli altri quattro schermi sono situati nel secondo lotto di 64K e sono chiamati Blocco 2, Blocco 3, Blocco 4 e Blocco 5.

Solo il Blocco 1 (dei primi 64K) può essere *visualizzato* sullo schermo. Per vedere uno schermo memorizzato nel secondo lotto di 64K (Blocco 2 fino a 5) è necessario spostare lo schermo richiesto nel Blocco 1. Il BANK MANAGER fornisce tutti i comandi necessari a spostare gli schermi come ad esempio il comando |SCREENCOPY che sposta uno schermo su un altro, sovrascrivendo in tal modo lo schermo su cui viene posto; e |SCREENSWAP che scambia i contenuti di due schermi.

BANK MANAGER utilizza i "comandi esterni" i quali iniziano con il simbolo di barra verticale (ottenuta premendo **[SHIFT]**@):

Come usare BANK MANAGER

Si riavvii il computer premendo **[CONTROL][SHIFT][ESC]**, si inserisca quindi il lato 1 (Size 1) del disco di sistema e si scriva:

RUN "BANKMAN"

La procedura di caricamento su RSX (Resident System eXtensions) è descritta in dettaglio nel Capitolo 7 parte 13; si consiglia, prima di usare tali routine all'interno di un programma, di conoscere la procedura RSX e lo spazio di memoria riservato. Comunque, per gli esempi che seguono, non è necessario conoscere la procedura di ca-ricamento.

Si scriva:

MODE 1

PRINT "QUESTO E' LO SCHERMO PER DIFETTO"
|SCREENSWAP,1,2

Il testo dovrebbe essere sparito; ciò che si vede sullo schermo è quello che è memorizzato come Schermo 2 (nel Blocco 2). Se il computer è stato appena acceso, ciò che si vede sullo schermo è stato determinato in modo casuale. Per calcellarlo si scriva:

MODE 1

PRINT "QUESTO E' LO SCHERMO 2"
|SCREENSWAP1,2

Riapparirà il testo iniziale. Se ora si ripete il comando |SCREENSWAP,1,2 si può vedere che il contenuto dei due schermi è cambiato. E' possibile vedere il contenuto di uno qualsiasi dei cinque schermi ma occorre ricordare che solo quando si fa una commutazione che implica il primo schermo la *visualizzazione* verrà modificata.

L'altro comando disponibile è |SCREENCOPY. Ciò permette di copiare uno schermo su un altro riscrivendo quest'ultimo con la nuova immagine.

Si scriva:

MODE 1

PRINT "QUESTO E' UNO SCHERMO DA COPIARE"
|SCREENCOPY,2,1

Il contenuto dello Schermo 1 viene copiato nello Schermo 2. Se si scambiano i parametri scrivendo:

```
MODE 1
|SCREENCOPY,1,2
```

...il contenuto dello schermo attualmente visualizzato viene sovrascritto dallo schermo 2.

Quindi il primo parametro è lo schermo su cui copiare; il secondo parametro è lo schermo da cui copiare.

In fase di copia degli schermi, si noti che si produrrà uno sfalsamento di visualizzazione se le modalità (MODE) degli schermi sono diverse, o se lo schermo si è spostato in sù di qualche riga dopo l'ultimo comando MODE. I comandi per lo schermo di BANK MANAGER sono stati progettati per essere usati con schermi grafici piuttosto che testi, è meglio quindi che che non accadano spostamenti di righe.

Uso di BANK MANAGER per la gestione delle stringhe

Vi sono altri quattro comandi di BANK MANAGER che permettono l'uso dei 64K di memoria addizionale come archivio di variabili stringa.

Molti programmi possono essere divisi in due parti: nella prima risiedono le istruzioni del programma, nella seconda i dati che il programma usa. Un ottimo esempio è un programma di archivio come ad esempio una agenda degli indirizzi. Un tale programma dovrebbe usare un vettore di stringhe per memorizzare i nomi e gli indirizzi delle persone presenti nel libro.

Le stringhe possono essere memorizzate nel secondo lotto di 64K di memoria in modo sequenziale fino alla fine. La memoria nella quale tali stringhe sono memorizzate può essere divisa in scomparti che prendono il nome di record. Il record può essere di una qualsiasi lunghezza fissata compresa tra 2 e 255 caratteri, dove la lunghezza di una stringa in BASIC varia a seconda del suo contenuto. Lo scopo del record è quello di fornire una dimensione standard agli scomparti nel quale memorizzare le stringhe di informazione. Ogni operazione di registrazione e ritrovamento dei dati da un record è seguita automaticamente da un passo nel record successivo, in attesa dell'operazione successiva. Il record da usare per l'operazione successiva è detto "record attuale" e verrà usato automaticamente a meno che non venga specificato uno differente.

Questo sistema di gestione di memoria viene chiamato da noi "disco RAM". Questo perchè si opera in modo analogo ai dischi ma usando la RAM.

Si leggano le descrizioni che seguono sui vari comandi per comprendere ciò che essi fanno e si guardino poi i diversi esempi.

Il primo comando del disco RAM è |BANKOPEN. Questo comando specifica quanti caratteri deve contenere ogni record. La sintassi è la seguente:

```
|BANKOPEN,n
```

dove il numero n specifica il numero di caratteri del record. Il numero n può assumere valori compresi tra 0 e 255, ma i valori 0 e 1 avranno strani effetti.

|BANKWRITE memorizza una stringa nel record attuale. Il record attuale viene quindi incrementato in modo che punti su quello successivo a quello appena scritto in attesa dell'operazione successiva. La sintassi è la seguente:

|BANKWRITE,@r%,a$

o

|BANKWRITE,@r%,a$,n

... dove r% è una variabile intera che restituisce un codice che fornisce informazioni riguardo l'operazione. a$ è una variabile stringa che contiene i caratteri che devono essere scritti nel record. Nel primo dei due esempi, il record è quello attuale; nel secondo, il parametro opzionale n specifica il record su cui scrivere.

|BANKREAD esamina il record attuale e restituisce il suo contenuto in una stringa. Il record attuale viene quindi incrementato in modo che punti su quello successivo a quello appena letto in attesa dell'operazione successiva. La lettura di un record non comporta la sua cancellazione; tale operazione può essere quindi ripetuta innumerevoli volte. La sintassi è la seguente:

|BANKREAD,@r%,a$

o

|BANKREAD,@r%,a$,n

... dove r% è una variabile intera che restituisce un codice che fornisce informazioni riguardo l'operazione. a$ è una variabile stringa che contiene i caratteri letti. Nel primo dei due esempi, il record è quello attuale; nel secondo, il parametro opzionale n specifica il record da leggere.

L'ultimo comando è |BANKFIND. Questo comando ricerca una particolare stringa tra i record. Se la stringa viene trovata restituisce il numero di record in cui è stata trovata. La sintassi è la seguente:

|BANKFIND,@r%,a$

o

|BANKFIND,@r%,a$,n

o

|BANKFIND,@r%,a$,n,m

... dove r% è una variabile intera che restituisce o il numero del record (dove è stata trovate la stringa) o un codice che indica che la stringa non è stata trovata. a$ è la stringa da cercare. Il parametro opzionale n indica il record da cui iniziare la ricerca: se non viene specificato la ricerca ha inizio dal record attuale. Il secondo parametro opzionale m specifica l'ultimo record da confrontare; la ricerca termina con tale record anche se infruttuosa. Se tale parametro viene omesso la ricerca continua fino alla fine di tutti i 64K continuando anche se i dati sono terminati.

Se si è già caricato il programma BANK MANAGER per vedere i comandi di schermo e non si è riavviato il computer allora i comandi sono già residenti in memoria. Se non lo si è fatto occorre inserire il lato 1 (Size 1) del disco di sistema e scrivere:

```
RUN "BANKMAN"
```

Si scriva ora:

```
|BANKOPEN,20
```

Questa istruzione fissa la lunghezza del record a 20 caratteri e il record attuale a 0.

Si scriva ora:

```
a$="PRIMA VOCE"+SPACE$(10)
```

...che pone a$ ad esattamente 20 caratteri.

Si scriva ora:

```
r%=0
```

... per inizializzare r%

Si scriva ora:

```
|BANKWRITE,@r%,a$
```

Questa istruzione scrive a$ nel primo record (record 0). Si scriva ora:

```
d$=SPACE$(20)
|BANKREAD,@r%,d$,0
PRINT d$
```

Il primo comando fa in modo che d$ contenga 20 spazi. Questo è uno spazio sufficiente a contenere i record da leggere. Il secondo comando legge il record numero 0 e pone il risultato in d$. Poichè il record attuale è il numero 1 (a causa della precedente operazione |BANKWRITE) è ora necessario specificare il record 0, quello che deve essere letto. Occorre ricordare che se non si specifica quale record leggere viene letto quello attuale. Al termine dell'operazione di lettura viene stampato il risultato. Quindi d$ dovrebbe contenere "PRIMA VOCE" e 10 spazi.

Si scriva ora:

```
b$="DUE"+SPACE$(17)
c$="TRE"+SPACE$(17)
|BANKWRITE,@r%,b$,1
|BANKWRITE,@r%,c$
```

Ciò pone b$ e c$ nei record 1 e 2. Anche in questo caso viene specificato, nel primo comando |BANKWRITE, il parametro opzionale chN specifica il record 1 come record nel quale porre b$. Il record attuale viene di conseguenza fissato automaticamente a 2; non è quindi necessario specificarlo.

Si scriva:

```
PRINT r%
```

Il risultato di questa istruzione dovrebbe essere 2. Questo numero può essere considerato come l'ultimo record su cui si è operato o come l'attuale record meno uno. Nell'esempio precedente l'ultimo record su cui si era operato era 2 e quello successivo 3.

Lo scopo di questo codice è quello di fornire informazioni sull'operazione effettuata. Una operazione conclusa con successo restituirà come codice un numero positivo che indica un record; un insuccesso restituirà un numero negativo. Vi sono due possibili errori che possono essere restituiti con |BANKWRITE e |BANKREAD. Essi sono:

-1 Indica che si è raggiunta la fine del file. Ciò accade quando si sono usati tutti i record o si è specificato un record inesistente.

-3 Indica un insuccesso nella commutazione di memoria. Ciò non dovrebbe mai accadere.

Si provino questi ulteriori esempi:

```
d$=STRING$(20,"X")
|BANKOPEN,20
FOR n=1 to 3:|BANKWRITE,@r%,d$:NEXT
```

d$ contiene ora 20 X; |BANKOPEN pone il record attuale a 0, quindi il comando IBANKWRITE sovrascriverà il contenuto dei record 0, 1 e 2 con d$.

Si scriva ora:

```
a$="PRIMO"
|BANKWRITE,@r%,a$,0
```

Ciò pone la parola "PRIMO" nel record 0 sovrascrivendo su alcune X. Ora si riempia ancora di 0 d$ mediante:

d$=SPACE$(20)

Si scriva ora:

|BANKREAD,@r%,d$,0

Questa istruzione riporta il record 0 in d$.

Riassumiamo i passi fatti fino ad ora.

Tutti e tre i record contengono X. Nel record 0 è stata aggiunta la parola "PRIMO". d$ è stato riempito di spazi ed infine il record 0 è stato portato in d$. Si scriva ora:

PRINT d$

Il risultato dovrebbe essere: "PRIMOXXXXXXXXXXXXXXX". Ciò illustra un'importante considerazione riguardo l'uso di questi comandi. Se la stringa posta nel record non lo riempie completamente i vecchi caratteri che non sono stati sovrascritti resteranno nel record. E' quindi utile porre nel record una stringa di spazi o di CHR$(32) prima di immettervi nuove informazioni. Questo per evitare di porre una stringa e trovare caratteri indesiderati. La stessa considerazione va fatta per stringhe nelle quali un record sarà letto. Se la stringa è maggiore della lunghezza del record alla fine della stringa vi saranno dei caratteri non modificati. Questo è il motivo per cui d$ è stato cancellato (riempito con spazi) prima di porvi il record 0.

E' possibile scrivere una stringa in un record troppo piccolo per accettare la stringa completa. Se ciò accade i caratteri eccedenti verranno ignorati.

E' possibile inoltre porre il contenuto di un record lungo in una stringa corta. Anche in questo caso i caratteri eccedenti (letti dal record) verranno ignorati. Nelle normali operazioni sulle stringhe, BASIC dovrebbe estendere automaticamente la stringa in modo che accetti i caratteri eccedenti ma ciò non accade con un comando esterno.

Vi è infine il comando |BANKFIND. Questo permette di cercare una particolare voce nei record. Se ad esempio il record numero 24 inizia con la parola "FED", è possibile ricercarla con il comando:

|BANKFIND,@r%,"FED"

Ciò è molto utile nei programmi archivio quando si deve ricercare un nome o un indirizzo.

|BANKFIND inizierà la ricerca dal record attuale e continuerà tale ricerca su tutti i record fino a quando non trova la stringa cercata o non raggiunge la fine del secondo lotto di 64K.

E' possibile specificare il numero di record dal quale iniziare la ricerca, aggiungendo tale numero alla fine del comando.

E' possibile includere, dopo l'inizio del numero di record, un ulteriore numero di record per specificare il record nel quale terminare la ricerca.

E' possibile includere |BANKFIND per cercare una stringa che non si trova all'inizio del record. A tal fine, i CHR$(0) vanno usati nella posizione del carattere precedente i caratteri significativi da ricercare. I CHR$(0) verranno trattati come "caratteri jolly" (come ? nei nomi di file di CP/M) che significa che vengono considerati come "qualsiasi carattere". Un esempio potrebbe essere:

```
a$=STRING$(10,0)+"FRED"
|BANKFIND,@r%,a$,0
```

Questo dovrebbe trovare la prima occorrenza della parola "FRED" (dovrebbe trovarsi tra l'undicesimo ed il quattordicesimo carattere). I primi dieci caratteri dovrebbero contenere un numero di telefono o una qualsiasi altra informazione che |BANKFIND dovrebbe ignorare.

Il numero di record corrispondente alla stringa trovata viene posto nella variabile intera r% usata come codice di ritorno (se la parola "FRED" è stata trovata), altrimenti il codice di ritorno conterrà -2.

Ulteriori dettagli...

Ulteriori informazioni riguardo BANK MANAGER si trovano nel Capitolo 7; si vedano inoltre le parti 13 e 14 del Capitolo 6 riguardanti l'uso delle estensioni RSX.

Ciò conclude la parte 11 del Corso Fondamenti sul 464/6128. Da ora in poi si dovrebbe conoscere ciò che la maggior parte dei tasti della tastiera fa e come usare alcuni dei più semplici comandi BASIC, come formattare un disco vergine in modo sia pronto per l'uso e come eseguire alcune delle più elementari funzioni sui dischi cioè LOAD, SAVE, CAT ecc oltre ad alcuni comandi AMSDOS, CP/M e BANK MANAGER.

I capitoli seguenti affrontano degli aspetti più specializzati di calcolo e del BASIC AMSTRAD.

Buona fortuna e buona lettura!

Capitolo 2
Dopo i Fondamenti...

Si è ora terminato di leggere il corso Introduttivo e si ha di fronte il computer acceso. Si è appena appreso come eseguire una operazione diverse volte usando il ciclo FOR NEXT, e come usare IF THEN per controllare se una condizione è vera.

Bene, si è scritto il nome dell'utente su tutto lo schermo; si vuole ora effettuare alcune operazioni: alcune utili, altre dilettevoli. Il capitolo che segue elenca le parole chiave di BASIC AMSTRAD a disposizione dell'utente, e specifica inoltre la descrizione della 'sintassi' ed il loro uso. Dopo ciò, quello che si può fare con il computer è limitato solo dalla propria immaginazione.

Se non si è mai usato un computer l'idea di 'programmazione' potrebbe intimorire. Non si abbia paura! E' più semplice di quanto non si creda e certamente è più semplice della conoscenza delle tecnologie e del 'gergo'. Non abbiate paura del BASIC; ben presto si troverà divertente la programmazione con questo linguaggio e si vedranno i frutti della propria fatica. La programmazione può essere un esercizio utile soprattutto quando si è principianti del computer e del linguaggio. Si ricordi che qualsiasi cosa si faccia occorre fare attenzione a non scrivere sui dischi originali di sistema CP/M: in questo caso nulla di quello che si scrive può danneggiare il computer ed è sempre utile provare qualche cosa di nuovo.

Da dove iniziare?

L'inizio è sempre la fase più difficile di un programma. Comunque, quello che occorre evitare è di buttarsi a capofitto e scrivere senza precedentemente pensare.

Una delle prime cose da stabilire è quello che il programma deve fare esattamente e come debbano essere presentati i risultati (in altre parole, come deve presentarsi lo schermo in fase di esecuzione).

Dopo aver deciso ciò si può iniziare a scrivere il programma che soddisfi le proprie richieste, e al contempo pensare a come rendere il programma fluido dall'inizio alla fine, con i minimi salti (GOTO). Un buon programma sarà di facile lettura e non comporterà confusione in fase di ricerca di errori.

Fortunatamente BASIC è un linguaggio indulgente ed aiuterà spesso l'utente visualizzando messaggi di errori quando quest'ultimo sbaglia. BASIC permette inoltre di avere 'ripensamenti' ed è possibile inserire una nuova linea tra quelle esistenti senza problemi.

Scriviamo un semplice programma.

Iniziamo dunque. Scriveremo un programma che tiene traccia dei numeri di telefono e degli indirizzi di amici. Applichiamo ora le regole precedenti: 'Cosa deve fare il programma?' e 'Come deve essere presentato il risultato?'.

Bene, il programma dovrebbe permettere di inserire e memorizzare ad esempio 100 nomi e numeri di telefono. Si deve poter poi inserire un nome ed ottenere il numero di telefono. In oltre, se non ci si ricorda del nome si dovrebbe essere in grado di visualizzare la lista completa. Si noti che in tal modo si è automaticamente considerato la domanda inerente la presentazione del risultato.

Bene, si mettano le dita sulla tastiera! Inizieremo ad inserire il titolo:

```
10 REM agenda telefonica
```

Non occorre porre un titolo nel programma, ma quando si inizia ad avere molti programmi è utile per sapere cosa fa.

Successivamente, sappiamo che vogliamo inserire una stringa di caratteri in una variabile; chiameremo tale variabile NOME$. La stessa considerazione vale per il numero di telefono; chiameremo tale variabile TEL$.

Ci si ricorda degli esempi del corso Fondamenti? Essi usavano il comando INPUT per inserire un valore della variabile, quindi possiamo scrivere:

```
20 INPUT"inserire il nome";NOME$
30 INPUT"inserire il numero di telefono";TEL$
run
```

E' possibile inserire il nome, ad esempio: Silvio e il numero di telefono, ad esempio 02 230222.

Il programma ha memorizzato l'informazione ma non ha prodotto nessun risultato sullo schermo. E' necessario del resto avere una parte di programma che permetta di recuperare l'informazione e visualizzarla. Per ottenere i valori di NOME$ e TEL$ dovremmo usare i comandi come:

```
PRINT NOME$  ...e... PRINT TEL$
```

Abbiamo detto che vogliamo essere in grado di memorizzare cento nomi e cento numeri di telefono. Sicuramente non dobbiamo scrivere un programma con duecento istruzioni INPUT, ognuna con una variabile diversa e quindi duecento istruzioni PRINT per produrre una lista sullo schermo? No, niente paura, il computer mette a disposizione quello che è noto come vettore proprio a tale scopo. Un vettore permette di usare una variabile (come NOME$) che può avere una qualsiasi dimensione (nel programma di esempio 100). Quindi, quando si vuole conoscere il contenuto di una variabile si usa il nome della variabile seguito dal suo numero di riferimento (racchiuso tra parentesi). Questo numero di riferimento è chiamato indice e l'espressione NOME$(27) è detta 'variabile indice'. Se si usa una variabile numerica come indice, per esempio NOME$(x) è possibile scandire tutta la lista di variabili, da 1 a 100, modificando il valore della x in un ciclo FOR NEXT; cioè FOR x=1 TO 100. Ogni volta che il valore della x aumenta di 1, il valore dell'indice cambia e fa riferimento ad un 'elemento' diverso o a un nome del vettore NOME$.

Vogliamo che i due vettori; uno NOME$ e l'altro TEL$ siano composti da 100 elementi. Prima di poter iniziare ad usare il vettore occorre dichiarare la DIMensione dei vettori. Riscriveremo le linee 20 e 30 con le seguenti:

```
20 DIM NOME$(100)
30 DIM TEL$(100)
```

Scriviamo ora una parte di programma che permetta di inserire e successivamente ritrovare i nomi ed i numeri. Si scriva:

```
40 FOR x=1 TO 100
50 INPUT;"  nome";NOME$(x)
60 INPUT;"  telefono";TEL$(x)
70 NEXT
run
```

Ciò va bene ma non si vogliono inserire tutti i 100 nomi in una sola volta. Inoltre, la presentazione stessa del programma sullo schermo è poco soddisfacente. Quello che occorre ora è un po' di ordine. Prima di inserire i dati è bene pulire lo schermo da tutto il testo superfluo. Si aggiunga:

```
45 CLS
```

Ora come indicare al computer che per il momento non si vogliono più inserire informazioni? Premendo **[ESC]** si ferma il programma ma quando lo si riesegue (tramite run) si sono persi tutti i valori inseriti nelle variabili!

Ecco cosa si può invece fare. Quando il programma prende un nuovo nome facciamo in modo che controlli che sia stato scritto qualcosa per il valore di NOME$(x), in altre parole controlla se il valore di NOME$(x) sia o meno la 'stringa vuota'. Se la stringa è vuota allora il programma smette di prendere valori. Si aggiunga:

```
55 IF NOME$(x)="" THEN 80
80 PRINT "fine inserimento"
```

Inoltre, per far sì che il programma indichi di terminare l'inserimento si aggiunga:

```
47 PRINT "Premere [RETURN] per terminare l'inserimento"
```

Si scriva ora la parte di programma che stampa l'informazione memorizzata sotto forma inizialmente di una lista. Si aggiunga:

```
90 FOR x=1 TO 100
100 PRINT NOME$(x);" ";TEL$(x)
110 NEXT
```

Ancora una volta il programma non si ferma fino a quando non ha raggiunto il centesimo elemento. Si aggiunga:

```
95 IF NOME$(x)="" THEN 120
120 PRINT "lista terminata"
```

La linea 95 scopre se NOME$(x) è una stringa vuota nel qual caso viene terminata la stampa e le linee 100 e 110 non vengono eseguite.

Scriviamo ora un programma che ricerchi un nome inserito. Si aggiunga:

```
130 INPUT "cerca";CERCA$
140 FOR x=1 TO 100
150 IF INSTR(NOME$(x),CERCA$)=0 THEN 180
160 PRINT NOME$(x);" ";TEL$(x)
170 END
180 NEXT
190 PRINT "nome non trovato"
run
```

Vi è un nuovo comando nella linea 150: INSTR. Questo indica al computer di INterrogare la prima espressione STRinga per cercare la prima occorrenza della seconda espressione stringa; in altre parole, cerca in NOME$(x) CERCA$ (che è la variabile inserita dall'utente nella linea 130 e che contiene il nome da cercare). Se INSTR non la trova restituisce il valore 0 che in questo caso viene usato per far saltare il programma nella linea 180 e cercare ancora nel valore successivo di x. Se il programma ha cercato tutti i 100 valori della x continua nella linea 190 ed indica che il nome non è stato trovato. Se comunque INSTR trova il nome non produce il valore 0 e salta alle linee 150 e 160 che visualizzano il nome ed il numero di telefono; il programma termina alla linea 170.

Come si è potuto vedere il programma si è evoluto in modo veloce, anche se comunque c'è ancora molto da fare. Riflettiamo un momento e consideriamo alcuni aspetti del programma iniziando dal modo in cui si esegue il programma: prima si scrive l'informazione, poi si ottiene la lista e quindi si cerca un nome specifico.

E se...?

E se non si vuole seguire quest'ordine? Si vuole ricercare un nome inserito ieri? Si vogliono aggiungere dei nomi e dei numeri di telefono a quelli già esistenti? Questi sono tutti gli aspetti del programma a cui pensare e trovare una soluzione; questo fa parte della programmazione. Come precedentemente menzionato, BASIC permette di aggiungere nuove istruzioni nel programma, ma un bravo programmatore deve aver previsto questi problemi.

Un altro dei maggiori problemi consiste nel fatto che tutti i valori delle variabili sono memorizzati in un'area di memoria che viene cancellata ogni qual volta si esegue un programma. Non si vorranno inserire tutte le informazioni ogni volta che si esegue il programma, quindi si vorranno salvare i valori contenuti nelle variabili NOME$ e TEL$ prima del termine del programma e li si vorranno caricare ogni volta che si esegue il programma.

Soluzioni

Il primo di questi problemi (l'ordine cioè con cui si eseguono le operazioni) può essere risolto scrivendo il programma in modo che permetta di scegliere al momento dell'esecuzione l'operazione che si desidera effettuare. Questo tipo di programma è detto 'guidato da menù' ed in effetti visualizza sullo schermo un menù dal quale selezionare una opzione. Se si è usato uno sportello automatico di una banca allora si è usato un programma guidato da menù! Aggiungiamo al programma tale menù:

```
32 PRINT "1. inserimento"
33 PRINT "2. elenco informazioni"
34 PRINT "3. ricerca"
35 PRINT "4. salvataggio informazioni"
36 PRINT "5. caricamento informazioni"
37 INPUT "inserire la scelta";ms
38 ON ms GOSUB 40,90,130

85 RETURN
125 RETURN
170 RETURN
200 RETURN
```

Come si è potuto vedere si è costruito il programma che stampa le opzioni del menù, chiede l'inserimento della scelta e pone quest'ultima nella variabile ms. Il comando ON ms GOSUB nella linea 38 indica al programma che se ms=1 questo deve saltare nella prima linea (40) della SUB-routine; se ms=2 allora deve andare nella seconda linea della SUB-routine (90) e così via.

Ognuna di queste funzioni richiamate dal comando ON ms GOSUB sono ora delle nuove routine e devono avere quindi alla fine il comando RETURN: è per questo motivo che li abbiamo aggiunti.

Ci si ricorda ciò che fa il comando RETURN ? Fa in modo che BASIC ritorni nel punto del programma seguente il comando GOSUB: in questo caso fa in modo che ritorni nella linea 38 (ciò significa che il programma continuerà alla linea 40: il punto di inserimento delle informazioni!). Non si vuole che ciò accada quindi si inserisce:

```
39 GOTO 32
```

...per far sì che il programma visualizzi ancora il menù. Si esegua (run) il programma: si potrà vedere che abbiamo fatto progressi.

Si guardi ora il listato del programma (se il programma è ancora in esecuzione si prema **[ESC]** due volte). Si scriva:

```
LIST
```

Dovrebbe apparire:

```
10 REM agenda telefonica
20 DIM NOME$(100)
30 DIM TEL$(100)
32 PRINT "1. inserimento"
33 PRINT "2. elenco informazioni"
34 PRINT "3. ricerca"
35 PRINT "4. salvataggio informazioni"
```

```
36 PRINT "5. caricamento informazioni"
37 INPUT "inserire la scelta";ms
38 ON ms GOSUB 40,90,130
39 GOTO 32
40 FOR x=1 TO 100
45 CLS
47 PRINT "Premere [RETURN] per terminare l'inserimento"
50 INPUT;"  nome";NOME$(x)
55 IF NOME$(x)="" THEN 80
60 INPUT;"  telefono";TEL$(x)
70 NEXT
80 PRINT "fine inserimento"
85 RETURN
90 FOR x=1 TO 100
95 IF NOME$(x)="" THEN 120
100 PRINT NOME$(x);" ";TEL$(x)
110 NEXT
120 PRINT "lista terminata"
125 RETURN
130 INPUT "cerca";CERCA$
140 FOR x=1 TO 100
150 IF INSTR(NOME$(x),CERCA$)=0 THEN 180
160 PRINT NOME$(x);" ";TEL$(x)
170 RETURN
180 NEXT
190 PRINT "nome non trovato"
200 RETURN
```

Si potrà notare che alcune linee hanno esaurito lo spazio per l'inserimento di altre linee, quindi per creare tale spazio e per mettere più ordine nel programma rinumeriamo le linee. Si scriva:

```
RENUM
LIST
```

Si deve ora vedere:

```
10 REM agenda telefonica
20 DIM NOME$(100)
30 DIM TEL$(100)
40 PRINT "1. inserimento"
50 PRINT "2. elenco informazioni"
60 PRINT "3. ricerca"
70 PRINT "4. salvataggio informazioni"
80 PRINT "5. caricamento informazioni"
90 INPUT "inserire la scelta";ms
```

```
100 ON ms GOSUB 120,210,270
110 GOTO 40
120 FOR x=1 TO 100
130 CLS
140 PRINT "Premere [RETURN] per terminare l'inserimento"
150 INPUT;"  nome";NOME$(x)
160 IF NOME$(x)="" THEN 190
170 INPUT;"  telefono";TEL$(x)
180 NEXT
190 PRINT "fine inserimento"
200 RETURN
210 FOR x=1 TO 100
220 IF NOME$(x)="" THEN 250
230 PRINT NOME$(x);" ";TEL$(x)
240 NEXT
250 PRINT "lista terminata"
260 RETURN
270 INPUT "cerca";CERCA$
280 FOR x=1 TO 100
290 IF INSTR(NOME$(x),CERCA$)=0 THEN 320
300 PRINT NOME$(x);" ";TEL$(x)
310 RETURN
320 NEXT
330 PRINT "nome non trovato"
340 RETURN
```

Così va meglio. Ed ora proseguiamo! Aggiungiamo ora una istruzione in modo che quando si inseriscono nuovi nomi e numeri di telefono il computer li aggiunga alle altre voci ponendole nel primo elemento del vettore che trova vuoto. Questa volta useremo un nuovo comando, ovvero LEN, che indica la lunghezza della stringa.

Si specificherà che se la lunghezza di NOME$(x) è maggiore di zero allora in quell'elemento è già presente una voce e quindi si farà saltare il programma alla linea 180 (che è un comando NEXT).

Coloro che hanno delle minime nozioni della lingua inglese troveranno molto facile comprendere le istruzioni di BASIC.

```
135 IF LEN(NOME$(X))>0 THEN 180
```

E' una soluzione semplice, vero? I problemi di questo tipo possono essere facilmente risolti utilizzano le parole chiave di BASIC ed un po' di intuito. C'è sempre un comando che può soddisfare le necessità dell'utente e più si programma più si troverà velocemente la soluzione.

Vediamo ora come salvare i contenuti delle variabili in modo che possano essere caricati quando si esegue il programma. Nella parte 7 del corso Introduttivo si è visto come salvare il programma stesso usando il comando SAVE. Comunque il programma stesso è solo una struttura che permette alle variabili di essere inserite (alla tastiera) e rilasciate (allo schermo). Quando si salva un programma si salva solo una struttura, non i valori delle variabili.

Ora dobbiamo scrivere una parte di programma che salva i valori delle variabili su disco. Facciamo ciò creando un 'file di dati' separato.

Prima di tutto apriamo il file e ne specifichiamo il nome "dati". Poi scriviamo i valori delle variabili NOME$ e TEL$ da 1 a 100 sul file ed infine chiudiamo il file e torniamo nel menù. Aggiungiamo tali istruzioni a partire dalla linea 350. Per non scrivere continuamente le linee del programma si usi il comando:

```
AUTO 350
```

... che fà si che le linee vengano incrementate automaticamente:

```
350 OPENOUT "dati"
360 FOR x=1 TO 100
370 WRITE #9,NOME$(x),TEL$(x)
380 NEXT
390 CLOSEOUT
400 PRINT "dati salvati"
410 RETURN
```

Dopo aver inserito la linea 410 e premuto il tasto **[RETURN]** si prema **[ESC]** per terminare la numerazione automatica delle linee.

Ora occorre inserire un ulteriore numero nella lista ON ms GOSUB della linea 100: questo perchè si è aggiunta un'altra opzione al menù da selezionare. Si editi quindi la linea 100 per aggiungere tale numero (EDIT 100):

```
100 ON ms GOSUB 120,210,270,350
```

Ora se si seleziona il numero 4 del menù il programma salverà tutte le informazioni inserite su disco.

Si noti nella linea 370, dove il programma indica di scrivere i valori di NOME$(x) e TEL$(x) su disco, l'espressione #9, usato dopo la parola WRITE. Il segno # è il segno di 'indicatore di canale'; in altre parole indica al computer a quale 'canale' inviare i dati. Il computer ha 10 canali.

Inviando i dati al canale da 0 a 7 (#0 a #7) questi appariranno sullo schermo poichè i canali #0 fino a #7 sono 'indicatori schermo' o finestre (WINDOW).

Inviando i dati al canale #8 si inviano i dati alla stampante (se connessa).

Infine, inviando i dati al canale #9 si inviano al drive, e ciò è quello che si è fatto nella linea 370.

Facendo un passo indietro...

Spendiamo ora poche parole riguardo il comando AUTO appena visto. Se si scrive:

AUTO

...il computer inizierà a numerare le linee cominciando dal numero 10 ed avanzando di 10 in 10 ad ogni pressione del tasto **[RETURN].** Se si sono usate precedemente le linee 10, 20, 30 ecc. il contenuto di ognuna di queste verrà visualizzato sullo schermo ad ogni pressione del tasto **[RETURN]**. Quando ogni linea appare sullo schermo questa può essere editata prima di premere **[RETURN]** fornendo così un metodo veloce di continuità di edizione sulle linee di programma successive.

Tornando al programma...

Si sono aggiunte le istruzioni per salvare le informazioni sul disco; l'ultima cosa che resta da fare è quella di caricare i dati ogni volta si esegue il programma. Occorre del resto aggiungere un'altra opzione di menù alla lista di numeri nella linea 100. Si editi nuovamente la linea 100 come segue:

```
100 ON ms GOSUB 120,210,270,350,420
```

Ora per caricare i dati occorre inserire delle nuove informazioni. Si inizia aprendo il file "dati" di ingresso (INPUT) del disco. Si prendono poi i dati dal disco (canale #9) di tutti i valori delle variabili NOME$(x) e TEL$(x) da 1 a 100 ed infine si chiude il file e si torna nel menù. Si scriva:

```
420 OPENIN "dati"
430 FOR x=1 TO 100
440 INPUT #9,NOME$(x),TEL$(x)
450 NEXT
460 CLOSEIN
470 PRINT "dati caricati"
480 RETURN
```

La fine dell'inizio...

Ora che si è scritto un programma che soddisfa le richieste iniziali tutto ciò che resta da fare è curare la visualizzazione sullo schermo.

L'inizio della fine...

Si aggiungano alcune istruzioni che rendono più ordinata la presentazione del programma:

```
34 MODE 1
```

Ciò stabilisce la modalità dello schermo e cancella lo schermo all'inizio del programma. Si aggiunga ora:

```
36 WINDOW #1,13,30,10,14
```

Non si creda che sia una istruzione molto complicata. Con tale istruzione si è creato sullo schermo una piccola finestra nella quale visualizzare il menù. Dopo la parola WINDOW si è specificato il numero di canale che la finestra deve usare; si ricorda che è possibile usare i canali da #0 a #7. Si deve a questo punto tenere in mente che tutto ciò che viene stampato sullo schermo usa il canale #0 a meno che non venga specificato altrimenti; non si vuole usare #0 per la piccola finestra in caso contrario tutto quello che il programma stampa viene inviato lì. Si specifica invece un altro canale, in questo caso si è usato #1. I quattro numeri che seguono #1 indicano la dimensione della finestra: i numeri specificano gli angoli sinistro, destro, in alto e in basso della finestra e fanno riferimento ai numeri di colonna e riga di testo (come il comando LOCATE). Quindi, nel precedente esempio il canale è #1, l'angolo sinistro della finestra inizia nella colonna 13, quello destro termina nella colonna 30, quello in alto inizia nella riga 10 e quello in basso termina nella riga 14.

Ora per far stampare nel canale #1 ciò che si è editato nelle linee dalla 40 alla 80 si modifichino tali linee come segue:

```
40 PRINT #1,"1. inserimento"
50 PRINT #1,"2. elenco informazioni"
60 PRINT #1,"3. ricerca"
70 PRINT #1,"4. salvataggio informazioni"
80 PRINT #1,"5. caricamento informazioni"
```

Si aggiunga ora:

```
85 LOCATE 7,25
```

Tale istruzione localizza il punto in cui l'istruzione INPUT della linea 90 apparirà in modo che risulti nitido.

Quindi, per far sì che lo schermo sia sempre pulito quando si visualizza il menù, si editi la linea 110 e la si modifichi come segue:

```
110 GOTO 34
```

Per far sì che lo schermo venga pulito dopo la scelta di ogni opzione si aggiunga:

```
95 CLS
```

Si aggiungano infine le seguenti tre linee di programma che fanno fare una pausa al programma prima di tornare nel menù:

```
103 LOCATE 9,25
105 PRINT "si prema un tasto qualsiasi per tornare al menù"
107 IF INKEY$="" THEN 107
```

La linea 103 indica al computer dove stampare il messaggio dato nella linea 105. La linea 107 chiede alla tastiera di controllare quale stringa variabile è stata inserita (mediante la pressione del tasto). Se scopre che la stringa è vuota (perchè non è stato premuto alcun tasto) allora il programma riesegue la stessa istruzione fino a quando INKEY$ non la trova più vuota. Ciò è utile quando si vuole inserire una pausa nel programma, in tal modo BASIC non passa a processare la linea seguente ma attende la pressione di un tasto.

Ora il programma è terminato. Si possono a questo punto aggiungere istruzioni per cancellare i nomi e i numeri di telefono, per ordinare alfabeticamente, per stampare l'elenco o, se si è veramente ambiziosi, per comporre automaticamente il numero dopo aver inserito il nome collegando (naturalmente con il permesso della SIP) il telefono al computer! Tutto ciò è possibile e si può continuare a testare ed ampliare il programma, soprattutto quando si ha a disposizione un computer potente come il 464/6128. Abbandoniamo ora questo programma sperando di aver appreso qualcosa in più sulla scrittura dei programmi. Si scriva ora, per rendere più chiaro il programma:

```
RENUM
```

...e lo si salvi su disco o lo si getti via. Potrebbe, comunque, servire a registrare i numeri di telefono dei propri amici!

Infine si listi il programma:

```
10 REM agenda telefonica
20 DIM NOME$(100)
30 DIM TEL$(100)
40 MODE 1
50 WINDOW #1,13,30,10,14
60 PRINT #1,"1. inserimento"
70 PRINT #1,"2. elenco informazioni"
80 PRINT #1,"3. ricerca"
90 PRINT #1,"4. salvataggio informazioni"
100 PRINT #1,"5. caricamento informazioni"
110 LOCATE 7,25
120 INPUT "inserire la scelta";ms
130 CLS
140 ON ms GOSUB 190,290,350,430,500
150 LOCATE 9,25
160 PRINT "si prema un tasto qualsiasi per tornare al menù"
170 IF INKEY$="" THEN 170
180 GOTO 40
190 FOR x=1 TO 100
200 CLS
210 IF LEN(NOME$(X))>0 THEN 260
220 PRINT "Premere [RETURN] per terminare l'inserimento"
230 INPUT;"  nome";NOME$(x)
240 IF NOME$(x)="" THEN 270
250 INPUT;"  telefono";TEL$(x)
260 NEXT
270 PRINT "fine inserimento"
280 RETURN
290 FOR x=1 TO 100
300 IF NOME$(x)="" THEN 330
310 PRINT NOME$(x);" ";TEL$(x)
320 NEXT
330 PRINT "lista terminata"
340 RETURN
350 INPUT "cerca";CERCA$
360 FOR x=1 TO 100
370 IF INSTR(NOME$(x),CERCA$)=0 THEN 400
380 PRINT NOME$(x);" ";TEL$(x)
390 RETURN
400 NEXT
410 PRINT "nome non trovato"
420 RETURN
430 OPENOUT "dati"
```

```
440 FOR x=1 TO 100
450 WRITE #9,NOME$(x),TEL$(x)
460 NEXT
470 CLOSEOUT
480 PRINT "dati salvati"
490 RETURN
500 OPENIN "dati"
510 FOR x=1 TO 100
520 INPUT #9,NOME$(x),TEL$(x)
530 NEXT
540 CLOSEIN
550 PRINT "dati caricati"
560 RETURN
run
```

Capitolo 3
Elenco completo delle parole chiave del BASIC del 464/6128

IMPORTANTE

E' fondamentale che si comprendano la terminologia e la notazione utilizzate in questo capitolo. Si incontreranno diversi tipi di parentesi, utilizzate durante la spiegazione di un comando che le contiene; ciascuna parentesi ha un significato specifico, come si potrà notare.

Ogni parte di un comando non presentata tra parentesi deve essere fornita così come viene data. Il comando END, ad esempio, avrà la forma:

END

e la parola END dovrà essere digitata interamente.

Quando un comando è incluso tra parentesi angolate <>, come ad esempio

<numero linea>

NON si devono digitare nè le parentesi, nè le parole in esse contenute: il precedente esempio mostra il tipo di dato richiesto nel comando. Ad esempio,

EDIT <numero linea>

significa che si dovrà digitare:

EDIT 100

Le parentesi tonde () DEVONO essere inserite letteralmente. Ad esempio,

COS(<espressione numerica>)

richiede l'inserimento delle parentesi intorno all'<espressione numerica> di cui è richiesto il COSeno, cioè:

PRINT COS(45)

Infine, le parentesi quadre racchiudono elementi opzionali in un comando o in una funzione. Ad esempio,

RUN [<numero linea>]

significa che non è necessario fornire un parametro alla parola chiave RUN, ma che è possibile espandere il comando inserendo il parametro opzionale <numero linea>. Quindi, il comando può essere digitato come:

RUN o RUN 100

Caratteri speciali

& o &H	Prefisso per costanti esadecimali
&X	Prefisso per costanti binarie
#	Prefisso per i canali

Tipi di dati

Le stringhe possono essere lunghe da 0 da 255 caratteri. Una <espressione di stringa> è una espressione che produce un valore di tipo stringa. Le stringhe possono essere concatenate l'una all'altra mediante l'operatore +, a condizione che la stringa risultante non sia maggiore di 255 caratteri.

I dati numerici possono essere interi o reali. Un dato intero è contenuto nell'intervallo -32768,.., 32767, mentre un dato reale è contenuto in una precisione leggermente superiore a nove cifre nell'intervallo ±1.7E+38, con il minimo valore al di sopra dello zero approssimativamente di 2.9E-39.

Una <espressione numerica> è una qualsiasi espressione il cui risultato sia un valore numerico. Possono essere semplici numeri, oppure una variabile numerica, oppure numeri utilizzati da variabili; si tratta di tutto ciò che non ricade sotto la categoria <espressione di stringa>.

Un <canale> si riferisce ad una <espressione numerica> che identifica una finestra sullo schermo, una stampante o un disco, dove è necessario inviare il testo.

Una <lista di:<elemento> descrive un parametro che comprende una lista di elementi separati da virgole. La lista può contenere un numero qualsiasi di elementi, limitati della lunghezza della linea.

Gli indicatori di tipo sono:

%	Intero
!	Reale (valore per difetto)
$	Stringa

Si osservi che le parole chiave del BASIC del 464/6128 AMSTRAD vengono presentate nella seguente forma:

PAROLA CHIAVE

Sintassi

Esempio

Descrizione

Parole chiave associate

Le parole chiave possono essere:

COMANDI : operazioni che vengono eseguite direttamente.
FUNZIONI : operazioni che vengono attivate come argomenti in una espressione.
OPERATORI : che agiscono su argomenti matematici.

BASIC converte tutte le parole chiave inserite in caratteri minuscoli in MAIUSCOLE nel LISTato di un programma. Gli esempi mostrati in questo capitolo utilizzano CARATTERI MAIUSCOLI, poichè il programma apparirà in questa forma quando verrà LISTato. Di conseguenza è preferibile inserirle in caratteri minuscoli, poichè risulterà semplice rilevare eventuali errori di battitura, in quanto le parole chiave verranno visualizzate in caratteri minuscoli in un LISTato.

Parole chiave...

ABS

ABS(<espressione numerica>)

```
PRINT ABS(-67.98)
 67.98
```

FUNZIONE: Restituisce il valore assoluto dell'espressione richiesta: perciò i numeri negativi vengono restituiti come positivi.

Parole chiave associate: SGN

AFTER

AFTER <ritardo di timer>[, <numero timer>] GOSUB <numero linea>

```
10 AFTER 250 GOSUB 60:CLS
20 PRINT "Indovina una lettera in 5 secondi"
30 a$=INKEY$:IF flag=1 THEN END
40 IF a$<>CHR$(INT(RND*26+97)) THEN 30
50 PRINT a$;" e' corretto. Hai vinto"
55 SOUND 1,478:SOUND 1,358:END
60 PRINT "Troppo tardi. Vinco io"
70 SOUND 1,2000:flag=1:RETURN
run
```

COMANDO: Richiama una subroutine BASIC dopo un dato periodo di tempo. Il parametro <ritardo di timer> specifica il periodo di tempo in unità di 0,02 (cinquantesimi di) secondo.

Il <numero timer> (compreso tra 0 e 3) specifica quale dei quattro timer deve essere usato. Il timer 3 ha priorità massima, mentre il timer 0 (quello per difetto) ha priorità minima.

Ciascun timer può avere una subroutine associata.

Ulteriori informazioni relative alle interruzioni si trovano nella parte 2 del capitolo "A vostra disposizione...".

Parole chiave associate: EVERY, REMAIN, RETURN.

AND

<argomento> AND <argomento>

```
IF "alan"<"bob" AND "gatto">"cane" THEN PRINT "corretto" ELSE PRINT "sbagliato"
corretto
IF "bob"<"alan" AND "cane">"gatto" THEN PRINT "corretto" ELSE PRINT "sbagliato"
sbagliato
IF "alan"<"bob" AND "cane">"gatto" THEN PRINT "corretto" ELSE PRINT "sbagliato"
sbagliato
....
PRINT 1 AND 1
 1
PRINT 0 AND 0
 0
PRINT 1 AND 0
 0
```

OPERATORE: Esegue operazioni booleane bit a bit su interi. Il risultato è 0 a meno che entrambi gli argomenti siano 1.

Ulteriori informazioni relative alla logica si trovano nella parte 2 del capitolo "A vostra disposizione...".

Parole chiave associate: OR, NOT, XOR.

ASC

ASC(<espressione di stringa>)

```
PRINT ASC("x")
 120
```

FUNZIONE: Restituisce il valore numerico del primo carattere in una <espressione di stringa>.

Parole chiave associate: CHR$.

ATN

ATN(<espressione numerica>)

```
PRINT ATN(1)
 0.785398163
```

FUNZIONE: Calcola l'ArcoTaNgente dell'<espressione numerica>.

Si osservi che si possono utilizzare DEG e RAD per ottenere il risultato rispettivamente in gradi o in radianti.

Parole chiave associate: COS, DEG, RAD, SIN, TAN.

AUTO

AUTO[<numero linea>] [,<incremento>]

```
AUTO 100,50
```

COMANDO: Genera AUTOmaticamente i numeri di linea. Il parametro opzionale <numero linea> fornisce la prima linea che deve essere generata nel caso si desideri generare linee da un particolare punto di un programma. Se il parametro è omesso, i numeri di linea verranno generati dalla linea 10 in poi.

L'<incremento> opzionale fornisce il numero di linee da lasciare prima di generare il successivo numero di linea. Se il parametro è omesso, i numeri di linea verranno incrementati ogni volta di 10.

Se viene generato un numero di linea già in uso, il contenuto della linea verrà visualizzato sullo schermo e potrà essere eventualmente modificato. La linea visualizzata verrà sostituita nella memoria quando viene premuto il tasto **[RETURN].**

Per annullare la numerazione automatica di linee si prema **[ESC].**

Parole chiave associate: **nessuna**.

BIN$

BIN$(<espressione intera senza segno>[,<espressione intera>])

```
PRINT BIN$(64,8)
01000000
```

FUNZIONE: Produce una stringa di cifre BINarie che rappresentano il valore dell'<espressione intera senza segno>, utilizzando il numero di cifre binarie specificate dalla seconda <espressione intera> (da 0 a 16). Se il numero di cifre specificate è eccessivo, l'espressione risultante verrà completata con l'inserimento all'inizio di alcuni 0; se il numero di cifre specificate non è sufficiente, l'espressione ottenuta NON verrà ridotta al numero specificato di cifre, ma sarà presentata con il necessario numero di cifre.

L'<espressione intera senza segno> che deve essere convertita in forma binaria deve avere un valore compreso tra -32768 e 65535.

Parole chiave associate: DEC$, HEX$, STR$.

BORDER

BORDER <colore>[,<colore>]

```
10 REM 729 combinazioni di cornice!
20 SPEED INK 5,5
30 FOR a=0 TO 26
40 FOR b=0 TO 26
50 BORDER a,b:CLS:LOCATE 14,13
60 PRINT "cornice";a;",";b
70 FOR t=1 TO 500
80 NEXT t,b,a
run
```

COMANDO: Modifica il colore della cornice dello schermo. Se vengono specificati due colori, la cornice si alternata tra i due secondo quanto specificato nel comando SPEED INK. I colori della cornice sono compresi tra 0 e 26.

Parole chiave associate: SPEED INK.

BREAK

(Si veda ON BREAK CONT, ON BREAK GOSUB, ON BREAK STOP).

CALL

CALL <espressione di indirizzo>[,<lista di parametri>]

CALL 0

COMANDO: Permette di richiamare da BASIC una subroutine sviluppata esternamente. La chiamata precedente riavvia il computer.

Non è un comando da usare con leggerezza.

Parole chiave associate: UNT.

CAT

CAT

COMANDO: CATalogo del disco e della cassetta

6128: Visualizza, in ordine alfanumerico, i nomi completi di tutti i file trovati, insieme alla loro lunghezza (arrotondata per eccesso al Kbyte più vicino). Viene inoltre visualizzato lo spazio libero rimasto sul disco, insieme all'identificazione dell'unità dischi e dell'utente.

464: Verrà indicato:

Press PLAY then any key

... a tal punto occorre premere il tasto **PLAY** del registratore e quindi un tasto qualsiasi del computer. Il nastro inizierà a girare ed il computer visualizzerà i nomi dei file presenti sulla cassetta.

Verrà visualizzato ogni blocco del file seguito da un carattere che indica il tipo di file:

$	Un file BASIC non protetto
%	Un file BASIC protetto
*	Un file ASCII
&	Un file binario

Alla fine della linea, se il computer ha letto il file visualizza il messaggio:

OK

La funzione CAT non modifica il programma in memoria.

Se un file è stato salvato senza nome CAT visualizza il messaggio:

Unnamed file
(File senza nome)

Premendo **[ESC]** si interrompe la CATalogazione della cassetta.

Parole chiave associate: LOAD, RUN, SAVE.

CHAIN

CHAIN <nome-file>[,<espressione di numero di linea>]

CHAIN "testprog.bas",350

COMANDO: Carica un programma in memoria da disco, sostituendo il programma esistente. Il nuovo programma inizia l'esecuzione, dall'inizio oppure da una linea specificata nell'espressione opzionale <espressione di numero di linea>.

I file protetti, (salvati con ,p) possono essere caricati ed eseguiti mediante il comando CHAIN.

464: Non è necessario specificare il nome del file su cassetta che si vuole caricare.

Verrà indicato:

Press PLAY then any key:

... a tal punto occorre premere il tasto **PLAY** del registratore e quindi un tasto qualsiasi del computer. Il nastro inizierà a girare ed il computer caricherà il file.

Lo schermo visualizzerà il seguente messaggio:

Loading NOMEFILE block 1

...e quindi aggiornerà il numero del blocco man mano che vengono caricati.

Se il primo carattere del nome del file è ! il messaggio precedente non verrà visualizzato e non sarà necessario premere un tasto (occorre ricordarsi però di premere il tasto **PLAY** del registratore). Se i propri programmi usano il ! ma possono operare anche su disco, in fase di lettura del disco il punto esclamativo verrà ignorato. Si noti che il ! NON occupa un carattere nel nome del file su cassetta o su disco.

L'abbandono del comando tramite il tasto [ESC] provoca la visualizzazione del seguente messaggio:

Broken in

Parole chiave associate: CHAIN, DELETE, LOAD, MERGE.

CHAIN MERGE

CHAIN MERGE <nome-file>[,<espressione di numero di linea>][,DELETE <blocco di linee>]

CHAIN MERGE "newrun.bas",750,DELETE 400-680

COMMAND: Carica un programma da disco, lo fonde con il programma attualmente in memoria e inizia ad eseguire il programma risultante, dall'inizio oppure da una linea specificata nell'<espressione di numero di linea> opzionale. Se prima di una operazione di CHAIN MERGE è necessario cancellare parte del programma originale, si può utilizzare l'opzione DELETE <blocco di linee>.

Si osservi che i numeri di linea del precedente programma che sono presenti anche nel nuovo programma su cui si esegue l'operazione di CHAIN MERGE verranno sostituiti dalle linee del nuovo programma.

I file protetti, (salvati con ,p) NON possono essere caricati ed eseguiti mediante il comando CHAIN MERGE.

464: Non è necessario specificare il nome del file su cassetta che si vuole caricare.

Verrà indicato:

Press PLAY then any key:

...a tal punto occorre premere il tasto **PLAY** del registratore e quindi un tasto qualsiasi del computer. Il nastro inizierà a girare ed il computer caricherà il file.

Lo schermo visualizzerà il seguente messaggio:

Loading NOMEFILE block 1

...e quindi aggiornerà il numero del blocco man mano che vengono caricati.

Se il primo carattere del nome del file è ! il messaggio precedente non verrà visualizzato e non sarà necessario premere un tasto (occorre ricordarsi però di premere il tasto **PLAY** del registratore). Se i propri programmi usano il ! ma possono operare anche su disco, in fase di lettura del disco il punto esclamativo verrà ignorato. Si noti che il ! NON occupa un carattere nel nome del file su cassetta o su disco.

L'abbandono del comando tramite il tasto [ESC] provoca la visualizzazione del seguente messaggio:

```
Broken in
```

Parole chiave associate: CHAIN, DELETE, LOAD, MERGE.

CHR$

CHR$(<espressione intera>)

```
10 FOR x=32 TO 255
20 PRINT x;CHR$(x),
30 NEXT
run
```

FUNZIONE: Converte una <espressione intera> tra 0 e 255 nel suo equivalente carattere, utilizzando l'insieme di caratteri AMSTRAD mostrato nella parte 3 del capitolo "A vostra disposizione".

Si noti che i caratteri da 0 a 31 sono caratteri di controllo; per questo motivo il precedente esempio stampa solo i caratteri compresi tra 32 e 255.

Parole chiave associate: ASC.

CINT

CINT(<espressione numerica>)

```
10 n=1.9999
20 PRINT CINT(n)
run
 2
```

FUNZIONE: Restituisce il valore dell'<espressione numerica>, Convertendola in un INTero compreso tra -32768 e 32767.

Parole chiave associate: CREAL, FIX, INT, ROUND, UNT.

CLEAR

CLEAR

```
CLEAR
```

COMANDO: Porta il valore di tutte le variabili a zero o nullo. Tutti i file aperti vengono abbandonati, tutte le funzioni utente vengono cancellate e BASIC è portato in modalità di calcolo in radianti.

Parole chiave associate: nessuna.

CLEAR INPUT

CLEAR INPUT

```
10 CLS
20 PRINT "Digita le lettere!"
30 FOR t=1 TO 3000
40 NEXT
50 CLEAR INPUT
run
```

COMANDO: Elimina tutti i caratteri inseriti in precedenza da tastiera che si trovano ancora nel buffer.

Per vedere gli effetti di questo comando, si esegua il precedente programma e si digitino le lettere quando richiesto. Quindi si cancelli la linea 50 del programma e si esegua nuovamente, notando la differenza.

Parole chiave associate: INKEY, INKEY$, JOY.

CLG

CLG[<inchiostro>]

```
LOCATE 1,20
CLG 3
```

COMANDO: Pulisce lo schermo grafico utilizzando il colore della carta. Se viene specificato l'<inchiostro>, lo schermo grafico viene portato a quel valore.

Parole chiave associate: CLS, GRAPHICS PAPER, INK, ORIGIN.

CLOSEIN

CLOSEIN

CLOSEIN

COMANDO: Chiude ogni file di input da disco (si veda OPENIN).

Parole chiave associate: EOF, OPENIN.

CLOSEOUT

CLOSEOUT

CLOSEOUT

COMANDO: Chiude ogni file di output su disco (si veda OPENOUT).

464: Se il buffer del file è in parte pieno e viene immesso il comando CLOSEOUT il computer salverà il contenuto del file rimanente su cassetta e verrà visualizzato il messaggio:

Press REC and PLAY then any key:

Lo schermo visualizzerà il seguente messaggio che indica che si sta salvando:

Saving NOMEFILE block

Se il primo carattere del nome del file nel comando OPENOUT corrispondente è ! il messaggio precedente non verrà visualizzato e non verrà richiesto di premere un tasto qualsiasi per salvare (occorre assicurarsi che i tasti **RECORD** e **PLAY** siano abbassati).

Se il primo carattere del nome del file è ! il messaggio precedente non verrà visualizzato e non sarà necessario premere un tasto nella tastiera (occorre ricordarsi però di premere il tasto **PLAY** del registratore). Se i propri programmi richiedono l'uso del ! questo in fase di operazione del disco verrà ignorato (quando il nome del file viene letto). Si noti che il ! NON occupa la posizione corrispondente ad un carattere nel nome del file su cassetta o nel disco.

L'abbandono del comando tramite il tasto [ESC] provoca la visualizzazione del seguente messaggio:

Broken in

Parole chiave associate: OPENOUT.

CLS

CLS[#<canale >]

```
10 PAPER#2,3
20 CLS#2
run
```

COMANDO: Pulisce il canale dello schermo (finestra) e lo riporta al suo colore. Se non viene fornito nessun <canale >, viene pulito il flusso sullo schermo (#0).

Parole chiave associate: CLG, INK, PAPER, WINDOW.

CONT

CONT

```
CONT
```

COMANDO: CONTinua l'esecuzione del programma, dopo una doppia pressione del tasto **[ESC]** oppure dopo aver incontrato un comando STOP nel programma. CONT prosegue l'esecuzione del programma se questo non è stato modificato oppure se non è protetto.

Prima della CONTinuazione possono essere inseriti comandi diretti.

Parole chiave associate: STOP.

COPYCHR$

COPYCHR$(#<canale >)

```
10 CLS
20 PRINT "angolo superiore"
30 LOCATE 1,1
40 a$=COPYCHR$(#0)
50 LOCATE 1,20
60 PRINT a$
run
```

FUNZIONE: Copia un carattere dalla posizione corrente del canale (che DEVE essere specificato). Il programma precedente copia un carattere dalla locazione 1, 1 (angolo superiore sinistro) e lo riproduce nella locazione 1, 20.

Se il carattere letto non viene riconosciuto, viene restituita una stringa nulla.

Parole chiave associate: LOCATE.

COS

COS(<espressione numerica>)

```
DEG
PRINT COS(45)
0.707106781
```

FUNZIONE: Calcola il COSeno dell'<espressione numerica>.

Si noti che si possono utilizzare DEG e RAD per ottenere il risultato dell'operazione rispettivamente in gradi o in radianti.

Parole chiave associate: ATN, DEG, RAD, SIN.

CREAL

CREAL(<espressione numerica>)

```
10 a=PI
20 PRINT CINT(a)
30 PRINT CREAL(a)
run
3
3.14159265
```

FUNZIONE: Restituisce il valore dell'<espressione numerica>, Convertendola in un numero REALe.

Parole chiave associate: CINT.

CURSOR

CURSOR[<interruttore di sistema>][,<interruttore utente>]

```
10 CURSOR 1
20 PRINT "domanda?"
30 a$=INKEYS:IF a$="" THEN 30
40 PRINT a$
50 CURSOR 0
run
```

COMANDO: Assegna al cursore il valore dell'interruttore di sistema o dell'interruttore utente, attivo o non attivo. I parametri <interruttore di sistema> e <interruttore utente> devono essere 0 (non attivo) o 1 (attivo). Nel precedente comando INKEY$, in cui il cursore non risulta generalmente visibile, il cursore sarà attivo in seguito all'assegnamento del valore 1 (nella linea 10) all'<interruttore di sistema>.

Il cursore viene visualizzato quando sia l'<interruttore di sistema> che l'<interruttore utente> sono attivi (hanno valore 1). L'<interruttore di sistema> viene automaticamente attivato per il comando INPUT, mentre è disattivato per INKEY$.

Si consiglia di disattivare il cursore quando si stampa testo sullo schermo.

E' possibile omettere uno dei due parametri singolarmente, ma non contemporaneamente. Se un parametro di interruttore viene omesso, quel particolare stato dell'interruttore non viene modificato.

Parole chiave associate: LOCATE.

DATA

DATA<lista di:<costante>

```
10 FOR x=1 TO 4
20 READ nome$,cognome$
30 PRINT nome$;" ";cognome$
40 NEXT
50 DATA Luigi, Rossi, Piero, Bianchi
60 DATA Sergio, Verdi, Alfredo, Testi
run
```

COMANDO: Dichiara i dati costanti che vengono usati in un programma. I dati possono essere inseriti nella variabile mediante il comando READ, dopo il quale il "puntatore" si sposta sul successivo elemento della lista DATA. Il comando RESTORE porta il puntatore ad una posizione specificata di DATA.

Per ulteriori informazioni relative ai dati si consulti la parte 2 del capitolo "A vostra disposizione...".

Parole chiave associate: READ, RESTORE.

DEC$

DEC$(<espressione numerica>,<maschera di formato>)

```
PRINT DEC$(10↑7,"££#########,.##")
 £10,000,000.00
```

FUNZIONE: Restituisce una stringa DECimale che rappresenta l'<espressione numerica>, utilizzando la <maschera di formato> specificata per controllare il formato di stampa della stringa ottenuta.

La maschera di formato può contenere SOLO i caratteri:

+ - £ $ * # , . ↑

L'uso di questi "specificatori di formato del campo" viene descritto sotto la parola chiave PRINT USING.

Parole chiave associate: BIN$, HEX$, PRINT USING, STR$.

DEF FN

DEF FN<nome funzione>[(<parametri formali>)]=<espressione>

```
10 t=TIME/300
20 DEF FNorologio=INT(TIME/300-t)
30 EVERY 100 GOSUB 50
40 GOTO 40
50 PRINT "Il programma e' stato eseguito";
60 PRINT FNorologio;"secondi fa"
70 RETURN
run
```

COMANDO: DEFinisce una FuNzione. BASIC permette al programma di definire e usare semplici funzioni che restituiscono valori. DEF FN è la parte di definizione di questo meccanismo e crea una funzione specifica del programma che opera all'interno del programma analogamente al modo in cui, ad esempio, opera COS come funzione predefinita di BASIC.

(Si noti nell'esempio precedente come il valore della funzione FNorologio venga continuamente aggiornato anche se il programma viene bloccato temporaneamente premendo **[ESC]**, oppure interrotto premendo due volte **[ESC]** e quindi riattivato con CONT).

Parole chiave associate: **nessuna.**

DEFINT

DEFINT <lista di:<insieme di lettere>

```
10 DEFINT n
20 numero=123.456
30 PRINT numero
run
 123
```

COMANDO: Assegna ad una variabile il tipo per DIFetto INTero. Quando si incontra una variabile senza un esplicito contrassegno di tipo (! % $), viene assunto il tipo per difetto. Questo comando assegna come tipo per difetto delle variabili con prima lettera specificata il tipo INTero. Può essere utilizzata una lista di lettere iniziali, come:

```
DEFINT a,b,c
```

oppure può essere presentato un insieme inclusivo di lettere iniziali come:

```
DEFINT a-z
```

Parole chiave associate: DEFREAL, DEFSTR.

DEFREAL

DEFREAL <lista di:<insieme di lettere>

```
DEFREAL x,a-f
```

COMANDO: Assegna ad una variabile il tipo per DIFetto REALe. Quando si incontra una variabile senza un esplicito contrassegno di tipo (! % $), viene assunto il tipo per difetto. Questo comando assegna come tipo per difetto delle variabili con prima lettera specificata il tipo REALe. Può essere utilizzata una lista di lettere iniziali, come:

```
DEFREAL a,b,c
```

oppure può essere presentato un insieme inclusivo di lettere iniziali come:

```
DEFREAL a-z
```

Parole chiave associate: DEFINT, DEFSTR.

DEFSTR

DEFSTR <lista di:<insieme di lettere>

```
10 DEFSTR nome
20 nome="Amstrad"
30 PRINT nome
run
 Amstrad
```

COMANDO: Assegna ad una variabile il tipo per DIFetto STRinga. Quando si incontra una variabile senza un esplicito contrassegno di tipo (! % $), viene assunto il tipo per difetto. Questo comando assegna come tipo per difetto delle variabili con prima lettera specificata il tipo STRinga. Può essere utilizzata una lista di lettere iniziali, come:

```
DEFSTR a,b,c
```

oppure può essere presentato un insieme inclusivo di lettere iniziali come:

```
DEFSTR a-z
```

Parole chiave associate: DEFINT, DEFREAL.

DEG

DEG

```
DEG
```

COMANDO: Attiva il modo di calcolo in gradi. La condizione per difetto delle funzioni SIN, COS, TAN e ATN è in radianti. DEG predispone BASIC all'uso dei gradi fino ad una istruzione RAD, oppure NEW, CLEAR, LOAD, RUN, ecc.

Parole chiave associate: ATN, COS, RAD, SIN, TAN.

DELETE

DELETE <intervallo di linee>

```
DELETE 100-200
```

COMANDO: Cancella parte del corrente programma, come specificato nell'espressione <intervallo di linee>.

E' possibile omettere il primo o l'ultimo numero dell'<intervallo di linee> per indicare ".. dall'inizio del programma" oppure ".. fino alla fine del programma", cioè:

```
DELETE -200
```

o

```
DELETE 50-
```

o

```
DELETE
```

per cancellare l'intero programma.

Parole chiave associate: CHAIN MERGE, RENUM.

DERR

DERR

```
LOAD "xyz.abc"

XYZ.ABC not found
Ready
PRINT DERR
 146
```

FUNZIONE: Restituisce l'ultimo codice di errore rilevato dal sistema di gestione di disco. Il valore DERR può essere utilizzato per controllare quale errore su disco si è verificato. Si consulti l'elenco dei messaggi di errore presente nel capitolo "A vostra disposizione".

Parole chiave associate: ERL, ERR, ERROR, ON ERROR GOTO, RESUME.

DI

DI

```
10 CLS:TAG:EVERY 10 GOSUB 90
20 X1=RND*320:X2=RND*320
30 Y=200+RND*200:C$=CHR$(RND*255)
40 FOR X=320-X1 TO 320+X2 STEP 4
50 DI
60 MOVE 320,0,1:MOVE X-2,Y:MOVE X,Y
70 PRINT " ";C$;:FRAME
80 EI:NEXT:GOTO 20
90 MOVE 320,0:DRAW X+8,Y-16,0:RETURN
run
```

COMANDO: Disabilita le Interruzioni (ad eccezione dell'interruzione **[ESC]**) finchè non vengono riabilitate esplicitamente da EI o implicitamente dal RETURN al termine di una subroutine di interruzione.

Si noti che l'ingresso in una subroutine di interruzione provoca la disabilitazione delle interruzioni di eguale o minore priorità.

Il comando viene usato per far eseguire il programma senza interruzioni: è il caso, ad esempio, di due routine interne ad un programma che competono per l'uso di risorse. Nel precedente esempio, il programma principale e la subroutine di interruzione competono per l'uso del monitor grafico.

Ulteriori informazioni relative alle interruzioni si trovano nella parte 2 del capitolo "A vostra disposizione".

Parole chiave associate: AFTER, EI, EVERY, REMAIN.

DIM

DIM <lista di:<variabile con indice>

```
10 CLS
20 DIM amico$(5),telefono$(5)
30 FOR n=1 TO 5
40 PRINT "Numero amico";n
50 INPUT "Inserisci nome";amico$(n)
60 INPUT "Inserisci n.tel.";telefono$(n)
70 PRINT
80 NEXT
90 FOR n=1 TO 5
100 PRINT n;amico$(n),telefono$(n)
110 NEXT
run
```

COMANDO: DIMensiona un vettore (array). DIM alloca lo spazio per i vettori e specifica il numero massimo di elementi. Si dovrà informare BASIC dello spazio che deve essere riservato per i vettori, altrimenti il valore per difetto sarà 10.

Un vettore è indentificato da una <variabile con indice>, dove un nome di variabile viene utilizzato con un insieme di numeri con indice, in modo che ogni "elemento" del vettore possieda il suo valore individuale. Il controllo del vettore può avvenire mediante cicli FOR NEXT, ad esempio, che possono esaminare il vettore, operando su ogni elemento.

Si noti che il minimo valore degli indici è zero (il primo elemento disponibile nel vettore).

I vettori possono essere multidimensionali, ed ogni elemento di un simile vettore viene rappresentato mediante la sua posizione all'interno della struttura del vettore. Se, ad esempio, il vettore è definito come:

```
DIM posizione$(20,20,20)
```

un elemento del vettore verrà rappresentato come:

```
posizione$(4,5,6)
```

Parole chiave associate: ERASE.

DRAW

DRAW <coordinata x>,<coordinata y>[,[<inchiostro>][,<modo inchiostro>]]

```
10 MODE 0:BORDER 0:PAPER 0:INK 0,0
20 x=RND*640:y=RND*400:z=RND*15
30 DRAW x,y,z
40 GOTO 20
run
```

COMANDO: Traccia una linea sullo schermo grafico, dalla posizione corrente del cursore fino alla posizione assoluta specificata dalle coordinate x e y. L'<inchiostro> con cui si disegna la linea può essere specificato (compreso tra i valori 0 e 15).

Il <modo inchiostro> opzionale determina il modo in cui la linea tracciata interagisce con quelle già presenti sullo schermo. I quattro <modi inchiostro> sono:

0: Normale
1: XOR (O esclusivo)
2: AND (E)
3: OR (O)

Parole chiave associate: DRAWR, GRAPHICS PEN, MASK.

DRAWR

DRAWR <spazio x>,<spazio y>[,[<inchiostro>][,<modo inchiostro>]]

```
10 CLS:PRINT "Vengo di sopra?"
20 MOVE 0,350:FOR n=1 TO 8
30 DRAWR 50,0
40 DRAWR 0,-50
50 NEXT:MOVE 348,0:FILL 3
60 GOTO 60
run
```

COMANDO: Traccia una linea sullo schermo grafico, dalla posizione corrente del cursore fino alla posizione relativa specificata da <spazio x e y>. L'<inchiostro> in cui si disegna la linea può essere specificato (compreso tra i valori 0..15).

Il <modo inchiostro> opzionale determina il modo in cui la linea tracciata interagisce con quelle già presenti sullo schermo. I quattro <modi inchiostro> sono:

0: Normale
1: XOR (O esclusivo)
2: AND (E)
3: OR (O)

Parole chiave associate: DRAW, GRAPHICS PEN, MASK.

EDIT

EDIT <numero linea>

    EDIT 20

COMANDO: Visualizza la linea di programma specificata in <numero linea> sullo schermo insieme al cursore, pronta per le modifiche.

Parole chiave associate: AUTO, LIST.

EI

EI

    EI

COMANDO: Riabilita le interruzioni che sono state disabilitate mediante il comando DI.

Se le interruzioni sono disabilitate in una subroutine di interruzione, vengono riabilitate automaticamente quando BASIC incontra il comando RETURN al termine della subroutine.

Ulteriori informazioni relative alle interruzioni si trovano nella parte 2 del capitolo "A vostra disposizione".

Parole chiave associate: AFTER, DI, EVERY, REMAIN.

ELSE

(Si consulti IF).

END

END

```
END
```

COMANDO: Termina l'esecuzione di un programma e ritorna al modo diretto. Un programma può contenere un numero arbitrario di comandi END, e viene automaticamente considerata la presenza di un END dopo la linea finale del programma.

Parole chiave associate: STOP.

ENT

ENT <numero inviluppo>[,<sezione inviluppo>][,<sezione inviluppo>][,<sezione inviluppo>][,<sezione inviluppo>][,<sezione inviluppo>]

```
10 ENT 1,10,-50,10,10,50,10
20 SOUND 1,500,200,10,,1
run
```

COMANDO: Assegna l'inviluppo di tono al <numero inviluppo> specificato (nell'intervallo da 1 a 15), che viene utilizzato dal comando SOUND. Se il <numero inviluppo> è negativo (nell'intervallo - 1.. -15), l'inviluppo viene ripetuto per tutta la durata del comendo SOUND.

Ciascuna <sezione inviluppo> può contenere 2 o 3 parametri.

Se vengono usati 3 parametri, questi sono:

<numero passi>,<dimensioni passi>,<tempo di pausa>.

Parametro 1: <numero passi>

Questo parametro specifica quanti diversi passi di tono devono essere attraversati durante la sezione inviluppo. In una sezione di una nota di durata 10 secondi, ad esempio, si possono desiderare 10 passi di tono di 1 secondo ciascuno. In questo caso il parametro <numero passi> usato dovrà essere 10.

L'intervallo disponibile di <numero passi> è da 0 a 239.

Parametro 2: <dimensioni passi>

Questo parametro deve essere nell'intervallo da -128 a +127. Passi negativi rendono più alto il tono della nota, mentre passi positivi lo rendono più basso. Il minimo periodo di tono è 0. L'elenco completo dei periodi di tono si trova nel capitolo "Riferimenti".

Parametro 3: <tempo di pausa>

Questo parametro specifica la durata di passi in unità di 0.01 secondi (centesimi di secondo). I valori del <tempo di pausa> si trovano nell'intervallo tra 0 e 255 (dove 0 viene considerato come 256), perciò il tempo maggiore tra passi è di 2,56 secondi.

Se vengono utilizzati due parametri, questi sono:

<periodo di tono>,<tempo di pausa>

Parametro 1: <periodo di tono>

Questo parametro fornisce un nuovo assegnamento assoluto per il periodo di tono (si veda il parametro 2 del comando SOUND).

Parametro 2: <tempo di pausa>

Questo parametro specifica la durata dei passi in unità di 0.01 secondi (centesimi di secondo). I valori del <tempo di pausa> si trovano nell'intervallo tra 0 e 255 (dove 0 viene considerato come 256), o 2,56 secondi.

In generale

Si noti che la lunghezza totale di tutti i <tempi di pausa> non deve risultare maggiore del parametro <durata> del comando SOUND, altrimenti il suono terminerà prima dell'esecuzione di tutti i passi di tono (in questo caso, il resto dell'inviluppo di tono viene eliminato).

Analogamente, se il parametro <durata> del comando SOUNO è più lungo della lunghezza totale di tutti i <tempi di pausa>, il suono proseguirà dopo l'esecuzione di tutti i passi di tono, rimanendo costante sull'ultimo tono.

In un comando ENT possono essere utilizzate fino a 5 <sezioni inviluppo> diverse (ciascuna composta dai precedenti 2 o 3 parametri).

Il primo passo di un inviluppo di tono viene eseguito immediatamente.

Ad ogni assegnamento di un inviluppo di tono, il valore precedente viene perso.

La specifica di un <numero inviluppo> senza <sezioni inviluppo> annulla ogni valore precedente.

Per maggiori informazioni relative ai suoni si consulti la parte 2 del capitolo "A vostra disposizione".

Parole chiave associate: ENV, SOUND.

ENV

ENV <numero inviluppo>[,<sezione inviluppo>][,<sezione inviluppo>][,<sezione inviluppo>][,<sezione inviluppo>][,<sezione inviluppo>]

```
10 ENV 1,15,-1,10,15,1,10
20 SOUND 1,200,300,15,1
run
```

COMANDO: Fissa l'inviluppo del volume specificato nel <numero inviluppo> (da 1 a 15) che viene usato dal comando SOUND.

Ciascuna <sezione inviluppo> può contenere 2 o 3 parametri.

Se vengono usati 3 parametri, questi sono:

<numero passi>,<dimensioni passi>,<tempo di pausa>.

Parametro 1: <numero passi>

Questo parametro specifica quanti diversi passi di volume devono essere attraversati dal suono durante la sezione inviluppo. In una sezione di una nota di durata 10 secondi, ad esempio, si possono desiderare 10 passi di volume di 1 secondo ciascuno. In questo caso il parametro <numero passi> usato dovrà essere 10.

L'intervallo disponibile di <numero passi> è da 0 a 127.

Parametro 2: <dimensioni passi>

Questo parametro può variare da un volume di livello 0 ad un volume di livello 15 relativamente al passo precedente. I 16 diversi livelli di volume sono gli stessi presenti nel comando SOUND, tuttavia il parametro <dimensioni passi> usato può assumere valori tra -128 e +127; il livello del volume ritorna ciclicamente a 0 ogni 15 unità.

Parametro 3: <tempo di pausa>

Questo parametro specifica il tempo tra passi in unità di 0.01 secondi (centesimi di secondo). I valori del <tempo di pausa> devono trovarsi nell'intervallo tra 0 e 255 (dove 0 viene considerato come 256), perciò il tempo maggiore tra passi è di 2,56 secondi.

Se vengono utilizzati due parametri, questi sono:

<inviluppo hardware>,<periodo inviluppo>

Parametro 1: <inviluppo hardware>

Questo parametro specifica il valore che deve essere inviato al registro di forma dell'inviluppo del chip sonoro.

Parametro 2: <periodo inviluppo>

Questo parametro specifica il valore che deve essere inviato al registro di periodo dell'inviluppo del chip sonoro.

Quando si usano inviluppi hardware si presuppone l'esistenza di una conoscenza hardware: in caso contrario, si suggerisce di usare un inviluppo software che comprenda un adatto parametro <tempo di pausa>.

In generale

Si noti che la lunghezza totale di tutti i <tempi di pausa> non deve risultare maggiore del parametro <durata> del comando SOUND, altrimenti il suono terminerà prima dell'esecuzione di tutti i passi di volume (in questo caso, il contenuto rimanente dell'inviluppo di volume viene eliminato).

Analogamente, se il parametro <durata> del comando SOUND è più lungo della lunghezza totale di tutti i <tempi di pausa>, il suono proseguirà dopo l'esecuzione di tutti i passi di volume, rimanendo costante sul livello finale.

In un comando ENV possono essere utilizzate fino a 5 <sezioni inviluppo> diverse (ciascuna composta da 2 o 3 parametri).

Il primo passo di un inviluppo di volume viene eseguito immediatamente.

Ad ogni assegnamento di un inviluppo di volume, il valore precedente viene perso.

La specifica di un <numero inviluppo> senza <sezioni inviluppo> annulla ogni valore precedente.

Per maggiori informazioni relative ai suoni si consulti la parte 2 del capitolo "A vostra disposizione".

Parole chiave associate: ENT, SOUND.

EOF

EOF

```
10 OPENIN "tasti.wp"
20 WHILE NOT EOF
30 LINE INPUT #9,a$
40 PRINT a$
50 WEND
60 CLOSEIN
run
```

FUNZIONE: Verifica se l'input da disco è alla fine del file. Restituisce -1 (vero) se nessun file è aperto o se il file è alla fine, altrimenti restituisce 0 (falso).

Parole chiave associate: OPENIN, CLOSEIN.

ERASE

ERASE <lista di:<nome variabile>

```
DIM a(100),b$(100)
ERASE a,b$
```

COMANDO: Cancella il contenuto di un vettore non più necessario, permettendo di utilizzare in altri modi la memoria.

Parole chiave associate: DIM.

ERL

ERL

```
10 ON ERROR GOTO 30
20 GOTO 1000
30 PRINT "errore nella linea";ERL
40 END
run
```

FUNZIONE: Restituisce il numero di linea dell'ultimo errore incontrato. Nel precedente esempio si vedrà che l'errore si trova nella linea 20, e questo viene rilevato dalla funzione ERL.

Parole chiave associate: DERR, ERR, ERROR, ON ERROR GOTO, RESUME.

ERR

ERR

```
GOTO 500
Line does not exist
Ready
PRINT ERR
 8
```

FUNZIONE: Restituisce il numero dell'ultimo ERRore incontrato. Si consulti l'elenco dei messaggi di errore riportato nel capitolo "Riferimenti". Nell'esempio precedente si vedrà che l'errore numero 8 corrisponde a "Line does not exist".

Parole chiave associate: DERR, ERL, ERROR, ON ERROR GOTO, RESUME.

ERROR

ERROR <espressione intera>

```
10 IF INKEY$="" THEN 10 ELSE ERROR 17
run
```

COMANDO: Richiama l'errrore specificato nell'<espressione intera>. Nel capitolo "Riferimenti" viene fornito un elenco dei messaggi di errore da 1 a 32. BASIC tratterà l'ERRORe come se fosse stato effettivamente rilevato e passerà direttamente ad una routine di gestione degli errori, oltre a restituire i valori di ERR e di ERL.

ERROR seguito da una <espressione intera> compresa tra 33 e 255 può essere utilizzato per creare messaggi di errore personalizzati, come nel seguente esempio:

```
10 ON ERROR GOTO 100
20 INPUT "inserisci un carattere";a$
30 IF LEN(a$)<>1 THEN ERROR 100
40 GOTO 20
100 IF ERR=100 THEN 110 ELSE 130
110 PRINT CHR$(7)
120 PRINT "Ho detto UN carattere!"
130 RESUME 20
run
```

Parole chiave associate: ERL, ERR, ON ERROR GOTO, RESUME.

EVERY

EVERY <periodo di tempo>[,<numero timer>] GOSUB <numero linea>

```
10 EVERY 50,1 GOSUB 30
20 GOTO 20
30 SOUND 1,20
40 RETURN
run
```

COMANDO: Richiama una subroutine BASIC ad intervalli regolari. Il <periodo di tempo> specifica l'intervallo in unità di 0.02 secondi (cinquantesimi di secondo).

Il <numero timer> (compreso tra 0 e 3) specifica quale dei quattro timer deve essere usato. Il timer 3 presenta la priorità più elevata; il timer 0 (quello per difetto) ha priorità minima.

E' possibile associare una sottoroutine ad ogni timer.

Per maggiori informazioni relative alle interruzioni si consulti la parte 2 del capitolo "A vostra disposizione".

Parole chiave associate: AFTER, REMAIN.

EXP

EXP(<espressione numerica>)

```
PRINT EXP(6.876)
968.743625
```

FUNZIONE: Calcola "E" elevato alla potenza specificata nell'<espressione numerica>, dove "E" è circa 2.7182818, cioè quel numero il cui logaritmo naturale è 1.

Parole chiave associate: LOG.

FILL

FILL <inchiostro>

```
10 MODE 0
20 FOR n=1 TO 500
30 PRINT "0"
40 NEXT
50 colorepenna=2+RND*13
60 FILL colorepenna
70 GOTO 50
run
```

COMANDO: Riempie un'area arbitraria dello schermo grafico. Gli estremi dell'area sono limitati da linee tracciate nel colore usato dalla penna grafica oppure nello stesso colore usato nell'area (compreso tra 0 e 15).

Il riempimento inizia dalla corrente posizione del cursore: se questa posizione corrisponde ad un estremo, non ci sarà riempimento.

Parole chiave associate: GRAPHICS PEN.

FIX

FIX (<espressione numerica>)

```
PRINT FIX(9.99999)
 9
```

FUNZIONE: Elimina la parte dell'<espressione numerica> posta a destra del punto decimale, arrotondandola verso lo zero.

Parole chiave associate: CINT, INT, ROUND.

FN

(Si consulti DEF FN).

FOR

FOR <variabile semplice>=<inizio> TO <fine>[STEP <passo>]

```
10 FOR n=2 TO 8 STEP 2
20 PRINT n;
30 NEXT n
40 PRINT ",chi ci piace?"
run
```

COMANDO: Esegue il corpo di un programma tra i comandi FOR e NEXT un certo numero di volte, incrementando la variabile di controllo tra un valore <inizio> e un valore <fine>. Se non viene specificato STEP <passo>, viene assunto 1.

Il valore di STEP <passo> può essere specificato come una <espressione numerica> negativa: in questo caso il valore del parametro di <inizio> deve essere maggiore del valore di <fine>, altrimenti la variabile di controllo non viene incrementata.

I cicli FOR NEXT possono essere "annidati", cioè uno può essere definito all'interno di un altro, a sua volta definito all'interno di un altro, e così via.

L'assegnamento del nome della variabile al comando NEXT è opzionale, poichè BASIC troverà automaticamente a quale comando FOR è associato un NEXT "anonimo".

Parole chiave associate: NEXT, STEP, TO.

FRAME

FRAME

```
10 MODE 0
20 PRINT "FRAME non attivo"
30 TAG
40 MOVE 0,200
50 FOR x=0 TO 500 STEP 4
60 IF f=1 THEN FRAME
70 MOVE x,200
80 PRINT " ";CHR$(143)
90 NEXT
100 IF f=1 THEN RUN
110 CLS
120 TAGOFF
130 PRINT "FRAME attivo"
140 f=1
150 GOTO 30
run
```

COMANDO: Sincronizza la scrittura di grafici sullo schermo insieme alla ricomposizione del video. L'effetto complessivo è che il movimento dei caratteri o della grafica risulta più dolce, senza effetti di scatti o lampeggi.

Parole chiave associate: TAG, TAGOFF.

FRE

FRE (<espressione numerica>)
FRE (<espressione di stringa>)

```
PRINT FRE(0)
PRINT FRE("")
```

FUNZIONE: Stabilisce quanta memoria libera rimane, non usata da BASIC. La forma FRE("") provoca un'operazione di pulizia della memoria ("garbage collection") prima di restituire un valore corrispondente allo spazio disponibile.

NOTA: BASIC utilizza solo 64K di memoria.

Parole chiave associate: HIMEM, MEMORY.

GOSUB

GOSUB <numero linea>

```
GOSUB 210
```

COMANDO: Esegue una subroutine BASIC portandosi direttamente al <numero linea> specificato. La fine della subroutine è contrassegnata dal comando RETURN, e dopo il suo raggiungimento il programma prosegue l'esecuzione dall'istruzione successiva al comando GOSUB.

Parole chiave associate: RETURN.

GOTO

GOTO <numero linea>

```
GOTO 90
```

COMANDO: Va ad un numero di linea specificato.

Parole chiave associate: nessuna.

GRAPHICS PAPER

GRAPHICS PAPER <inchiostro>

```
10 MODE 0
20 MASK 15
30 GRAPHICS PAPER 3
40 DRAW 640,0
run
```

COMANDO: Assegna l'"inchiostro" alla carta grafica, cioè alla zona posta dietro i grafici presenti sullo schermo. Quando vengono tracciate linee continue, la carta grafica non viene vista. Nell'esempio precedente, il comando MASK abilita la realizzazione di una linea e la visualizzazione della carta grafica.

L'inchiostro della carta grafica (compreso tra i valori 0 e 15) viene usato per la zona di "carta" dove vengono scritti i caratteri quando TAG è attivo, e come valore per difetto quando si pulisce la finestra grafica, mediante CLG.

Parole chiave associate: CLG, GRAPHICS PEN, INK, MASK, TAG, TAGOFF.

GRAPHICS PEN

GRAPHICS PEN [<inchiostro>][,<modo sfondo>]

```
10 MODE 0
20 GRAPHICS PEN 15
30 MOVE 200,0
40 DRAW 200,400
50 MOVE 639,0
60 FILL 15
run
```

COMANDO: Assegna il valore di <inchiostro> (compreso tra 0 e 15) usato per tracciare linee e punti. Il <modo sfondo> grafico può inoltre assumere i valori:

0: Sfondo opaco
1: Sfondo trasparente.

(Lo sfondo trasparente ha effetti sulla carta grafica dei caratteri scritti con TAG attivo, e gli intervalli nelle linee tratteggiate).

E' possibile omettere i due parametri, ma non entrambi contemporaneamente. Se un parametro viene omesso, quel valore non viene modificato.

Parole chiave associate: GRAPHICS PAPER, INK, MASK, TAG, TAGOFF.

HEX$

HEX$ (<espressione intera senza segno>[,<dimensione campo>])

```
PRINT HEX$(255,4)
00FF
```

FUNZIONE: Produce una stringa di cifre esadecimali che rappresentano il valore dell'<espressione intera senza segno>, utilizzando il numero di cifre esadecimali specificate nella <dimensione campo> (compresa tra 0 e 16). Se il numero di cifre specificato è troppo elevato, l'espressione risultante verrà completata con l'inserimento all'inizio di zeri; se il numero di cifre specificato non è sufficiente, l'espressione ottenuta NON verrà abbreviata fino al numero di cifre specificato, ma verrà presentata nel numero di cifre necessarie.

L'<espressione intera senza segno> che deve essere convertita in forma esadecimale deve presentare un valore compreso tra -32768 e 65535.

Parole chiave associate: BIN$, DEC$, STR$, UNT.

HIMEM

HIMEM

```
PRINT HIMEM
42619
```

FUNZIONE: Restituisce l'indirizzo del più elevato byte di memoria usato da BASIC (che può essere alterato dal comando MEMORY).

NOTA: BASIC utilizza solo 64K di memoria.

Parole chiave associate: FRE, MEMORY, SYMBOL, SYMBOL AFTER.

IF

IF <espressione logica> THEN <parte opzione> [ELSE <parte opzione>]

```
10 MODE 1
20 x=CINT(RND*100)
30 PRINT "Indovina il numero (da 0 a 100)"
40 INPUT n
50 IF n<x THEN PRINT n;"è troppo basso"
60 IF n>x THEN PRINT n;"è troppo alto"
70 IF n=x THEN 80 ELSE c=c+1:GOTO 40
80 PRINT "Benissimo, l'hai centrato!"
90 PRINT c+1;"tentativi!"
run
```

COMANDO: Determina se l'<espressione logica> è vera, e in questo caso esegue la prima <parte opzione>. Se l'<espressione logica> è falsa, viene eseguita la <parte opzione> specificata nella clausola ELSE, altrimenti BASIC passa alla linea successiva.

I comandi IF THEN possono essere annidati ad una qualsiasi profondità e sono conclusi dalla fine della linea. Di conseguenza NON è possibile avere altre istruzioni indipendenti dal comando IF THEN sulla stessa linea.

Quando il risultato dell'<espressione logica> richiede di saltare ad una linea particolare, il comando può assumere la forma seguente.

Esempi:

```
IF a=1 THEN 100
```

oppure

```
IF a=1 GOTO 100
```

oppure

```
IF a=1 THEN GOTO 100
```

Parole chiave associate: ELSE, GOTO, THEN.

INK

INK <inchiostro>,<colore>[,<colore>]

```
10 MODE 1:PAPER 0:PEN 1
20 FOR p=0 TO 1
30 FOR i=0 TO 26
40 INK p,i
50 LOCATE 16,12:PRINT "inchiostro";p;",";i
60 FOR t=1 TO 400:NEXT t,i,p
70 INK 0,1:INK 1,24:CLS
run
```

COMANDO: Assegna un colore (o più colori) ad un inchiostro specificato. Il parametro <inchiostro> rappresenta il riferimento all'inchiostro, che deve essere un'espressione intera da 0 a 15, per poter essere utilizzata nel corrispondente comando PEN o PAPER. Il primo parametro <colore> deve essere un'espressione intera con valore di colore compreso tra 0 e 26. Se viene specificato un secondo <colore> opzionale, l'inchiostro viene alternato tra i due colori, con una frequenza determinata dal comando SPEED INK.

Parole chiave associate: GRAPHICS PAPER, GRAPHICS PEN, PAPER, PEN, SPEED INK.

INKEY

INKEY (<espressione intera>)

```
10 IF INKEY(55)<>32 THEN 10
20 PRINT "Hai premuto [SHIFT] e V"
30 CLEAR INPUT
run
```

FUNZIONE: Interroga la tastiera per conoscere i tasti che sono stati premuti. La tastiera viene esaminata ogni 0.02 secondi (cinquantesimi di secondo).

La funzione è utile per controllare se un particolare tasto viene tenuto premuto o meno, rilevando il valore restituito -1 (che si verifica a prescindere dallo stato dei tasti **[SHIFT]** e **[CONTROL]**).

L'esempio precedente rileva quando i tasti **[SHIFT]** e **V** (tasto numero 55) sono premuti insieme, quindi termina il programma. Una spiegazione dei numeri dei tasti si trova nello schema sulla parte superiore destra del computer e nel capitolo "Riferimenti".

Lo stato di **[SHIFT]** e **[CONTROL]** insieme al tasto specificato nell'<espressione intera> viene identificato nel modo seguente:

Valore restituito	**[SHIFT]**	**[CONTROL]**	tasto specificato
-1	SU/GIU	SU/GIU	SU
0	SU	SU	GIU
32	GIU	SU	GIU
128	SU	GIU	GIU
160	GIU	GIU	GIU

Parole chiave associate: CLEAR INPUT, INKEY$, JOY.

INKEY$

INKEY$

```
10 CLS
20 PRINT "Selezioni Si o No (S/N)?"
30 a$=INKEY$
40 IF a$="" THEN 30
50 IF a$="s" OR a$="S" THEN 80
60 IF a$="n" OR a$="N" THEN 90
70 GOTO 30
80 PRINT "Hai selezionato Si":END
90 PRINT "Hai selezionato No"
run
```

FUNZIONE: Interroga la tastiera e restituisce il valore della stringa corrente corrispondente al tasto premuto. Se non vengono premuti tasti, INKEY$ restituisce una stringa vuota. Nell'esempio precedente, le linee 40 e 70 indicano al programma di ritornare alla linea 30 dopo aver interrogato la stringa ricevuta dalla tastiera.

Parole chiave associate: CLEAR INPUT, INKEY.

INP

INP(<numero porta>)

```
PRINT INP(&FF77)
 255
```

FUNZIONE: Restituisce il valore in ingresso dall'indirizzo di I/O specificato in <numero porta>.

Parole chiave associate: OUT, WAIT.

INPUT

INPUT[#<canale>,][;][<stringa con apice><separatore>]<lista di:<variabile>

```
	10 MODE 1
	20 INPUT "Dammi due numeri da moltiplicare (separati da una virgola)";a,b
	30 PRINT a;"per";b;"e'";a*b
	40 GOTO 20
	run
```

COMANDO: Accetta dati in ingresso dal canale specificato (se non specificato, dal canale #0).

Il primo punto e virgola [;], opzionale, elimina il carattere di ritorno carrello/nuova linea che viene altrimenti inserito dopo l'esecuzione del comando.

Il <separatore> deve essere un punto e virgola o una virgola. Un punto e virgola provoca la visualizzazione di un punto di domanda; una virgola elimina il punto di domanda.

Se viene inserito un dato di tipo errato, come la lettera O invece di 0 (zero), quando si digitano variabili numeriche, BASIC risponde con:

```
	?Redo from start (Ripeti dall'inizio)
```

e visualizza quindi il testo di prompt previsto.

Tutte le risposte date da tastiera devono essere concluse premendo il tasto **[RETURN]**.

Parole chiave associate: LINE INPUT.

INSTR

INSTR([<posizione inizio>,]<espressione esaminata>,<stringa cercata>)

```
10 CLS
20 alfabeto$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
30 INPUT "Inserisci una lettera";a$
40 b$=UPPER$(a$)
50 PRINT b$;" è un numero";
60 PRINT INSTR(alfabeto$,b$)
70 PRINT "nell'alfabeto.":PRINT
80 GOTO 40
run
```

FUNZIONE: Esamina la prima <espressione esaminata> per trovare la <stringa cercata> e restituisce la posizione della sua prima occorrenza all'interno dell'<espressione esaminata>. Se la <stringa cercata> non si trova nell'<espressione esaminata>, viene restituito 0.

La posizione da cui iniziare la ricerca della stringa può essere specificata mediante il parametro opzionale <posizione inizio>, che deve contenere un numero intero compreso tra 1 e 255.

Parole chiave associate: **nessuna**.

INT

INT(<espressione numerica>)

```
PRINT INT(-1.995)
-2
```

FUNZIONE: Arrotonda il numero al minore INTero vicino, eliminando la parte frazionaria. Nel caso di valori positivi restituisce lo stesso valore di FIX, mentre per i valori negativi non interi restituisce un numero in meno rispetto a FIX.

Parole chiave associate: CINT, FIX, ROUND.

JOY

JOY(<espressione intera>)

```
10 PRINT “Per fermare il programma -”;
20 PRINT “usare il joystick”
30 IF JOY(0)<>0 THEN END
40 GOTO 10
run
```

FUNZIONE: Legge un risultato da leggere da bit a bit dal JOYstick specificato nell'<espressione intera>.(0 o 1).

Bit	Decimale
0:Su	1
1:Giù	2
2:Sinistra	4
3:Destra	8
4:Fuoco 2	16
5:Fuoco 1	32

Quindi, ad esempio, se il pulsante principale di “fuoco” (Fuoco 2) del primo joystick viene premuto mentre l'impugnatura viene spostata verso sinistra, la funzione JOY(0) restituisce il valore decimale 20, corrispondente a 16 (Fuoco 2) + 4 (Sinistra).

Maggiori informazioni relative ai joystick si trovano nel capitolo “Riferimenti”.

Parole chiave associate: CLEAR INPUT, INKEY.

KEY

KEY <numero di espansione simbolo>,<espressione di stringa>

```
KEY 11,"border 13:paper 0:pen 1:ink 0,13:
ink 1,0:mode 2:list"+CHR$(13)
```

si prema ora il tasto **[IRETURN]**.

COMANDO: Assegna l'<espressione di stringa> al <numero di espansione simbolo> specificato. Possono essere gestiti trentadue simboli di espansione, compresi tra 0 e 31, corrispondenti ai valori da 128 a 159. I tasti da 128 (0 sul tastierino numerico) a 140 (**[CONTROL][RETURN]**) vengono assegnati per difetto alla stampa dei numeri da 0 a 9, di un punto decimale, di **[RETURN]** e di RUN"**[RETURN]** - (per operazioni su cassette), ma possono essere riassegnati ad altre <espressioni di stringa> . I simboli di espansione da 13 a 31 (valori da 141 a 159) sono stringhe normalmente vuote, ma possono essere espansi e assegnati a tasti, utilizzando il comando KEY DEF descritto nell'esempio seguente.

Il <numero di espansione simbolo> dato nel comando KEY deve essere compreso tra 0 e 31, o eventualmente tra 128 e 159 per avere una corrispondenza con i valori dei tasti (si faccia riferimento allo schema dei tasti nel capitolo "Riferimenti").

Nell'<espressione di stringa> può essere espanso un totale di 120 caratteri. Tentativi di maggiori espansioni provocano un errore(5) "Improper argument".

Parole chiave associate: KEY DEF.

KEY DEF

KEY DEF <numero tasto>,<ripete>[,<normale>,[<maiuscolo>[,<control>]]]

```
KEY 159,"questo e' il tasto Tab"
KEY DEF 68,1,159
```

si prema ora il tasto **[TAB].**

COMANDO: Definisce i valori dei tasti che devono essere restituiti dal <numero tasto> specificato, compresi tra 0 e 79 (per uno schema dei valori dei tasti si consulti il diagramma nella parte superiore destra del computer, oppure il capitolo "Riferimenti"). I parametri <normale>, <maiuscolo> e <control> devono contenere i valori da restituire quando vengono premuti, rispettivamente, da soli, insieme a **[SHIFT]** o insieme a **[CONTROL].** Ciascun parametro è opzionale.

Il parametro <ripete> permette di assegnare un valore attivo-non attivo (1 o 0) alla funzione di auto ripetizione del tasto, regolando la velocità di auto ripetizione mediante il comando SPEED KEY.

Nell'esempio precedente al tasto 159 (equivalente al simbolo di espansione 31) viene assegnata innanzitutto una stringa di espansione, quindi il comando DEF KEY definisce il tasto 68 (il tasto **[TAB]**) per auto ripetizione (1) e il suo ritorno al valore <normale> 159 quando è premuto da solo.

Nell'esempio prcedente, l'azione normale verrebbe ripristinata mediante:

```
KEY DEF 68,0,9
```

dove 9 è il normale valore ASCII per il carattere di tabulazione **[TAB]**.

Parole chiave associate: KEY, SPEED KEY.

LEFT$

LEFT$(<espressione di stringa>,<lunghezza richiesta>)

```
10 CLS
20 a$="AMSTRAD"
30 FOR n=1 TO 7
40 PRINT LEFT$(a$,n)
50 NEXT
run
```

FUNZIONE: Restituisce il numero di caratteri (compresi tra 1 e 255) specificato nel parametro <lunghezza richiesta>, dopo averli estratti dalla sinistra (LEFT) dell'<espressione di stringa>. Se l'<espressione di stringa> è minore della <lunghezza richiesta>, viene restituita tutta l'<espressione di stringa>.

Parole chiave associate: MID$, RIGHT$.

LEN

LEN(<espressione di stringa>)

```
10 LINE INPUT "Inserisci una frase";a$
20 PRINT "La frase e' lunga";
30 PRINT LEN(a$);"caratteri."
run
```

FUNZIONE: Restituisce il numero totale di caratteri (la lunghezza) dell'<espressione di stringa>.

Parole chiave associate: **nessuna**.

LET

LET <variabile>=<espressione>

```
LET x=100
```

COMANDO: Assegna un valore ad una variabile. E' un residuo dei primi BASIC in cui si dovevano vedere gli assegnamenti alle variabili. Non viene utilizzato nel BASIC AMSTRAD, e serve solo per assicurare la compatibilità con i programmi in precedenti BASIC forniti insieme ai manuali di istruzioni. L'esempio precedente può essere trasformato semplicemente in:

```
x=100
```

Parole chiave associate: **nessuna**

LINE INPUT

LINE INPUT [#<canale>,][;][<stringa tra apici><separatore>]<variabile di stringa>

```
10 LINE INPUT "Inserisci una linea di testo";a$
20 CLS
30 PRINT "La variabile a$ e' uguale a:-"
40 PRINT a$
run
```

COMANDO: Accetta in ingresso un'intera linea di testo dal canale indicato (o dal canale #0 se non specificato). Il primo punto e virgola [;] opzionale elimina il carattere di ritorno carrello/nuova linea che viene altrimenti inserito dopo l'esecuzione del comando.

Il <separatore> deve essere un punto e virgola o una virgola. Un punto e virgola provoca la visualizzazione di un punto di domanda; una virgola elimina il punto di domanda.

LINE INPUT dalla tastiera viene terminato dalla pressione del tasto **[RETURN]**.

LINE INPUT dal canale di disco (o di cassetta) #9 viene terminato dalla ricezione di un carattere di ritorno carrello o dall'assegnazione di 255 caratteri alla <variabile di stringa>, in base alla prima delle condizioni che si verifica.

Parole chiave associate: INPUT.

LIST

LIST [<numeri di linea>][,#<canale>]

```
LIST 100-1000,#1
```

COMANDO: Elenca linee di programma sul canale specificato. Il canale #0 è il canale di schermo, assunto per difetto, mentre #8 è la stampante. L'operazione può essere interrotta temporaneamente premendo una volta **[ESC]** e ripristinata con la pressione della barra spaziatrice. Premendo **[ESC]** due volte si conclude l'operazione, ritornando al modo diretto.

Il primo o l'ultimo numero del parametro <numeri di linea> possono essere omessi per indicare "dall'inizio del programma" o "fino alla fine del programma", cioè:

```
LIST -200
```

oppure

```
LIST 50-
```

oppure

```
LIST
```

per ottenere il listato dell'intero programma.

Parole chiave associate: **nessuna.**

LOAD

LOAD <nomefile>[,<espressione di indirizzo>]

```
LOAD "filedisc.xyz",&2AF8
```

COMANDO: Carica un programma BASIC da disco in memoria, sostituendo i programmi eventualmente presenti. Specificando l'<espressione di indirizzo> opzionale si otterrà il caricamento del file binario a quell'indirizzo, invece dell'indirizzo in cui era stato salvato.

Un programma BASIC protetto NON può essere caricato mediante il comando LOAD, perchè verrebbe immediatamente cancellato dalla memoria. Si usino in questo caso i comandi RUN o CHAIN.

464: Non è necessario specificare il nome del file su cassetta che si vuole caricare.

Verrà indicato:

```
Press PLAY then any key:
```

...a tal punto occorre premere il tasto **PLAY** del registratore e quindi un tasto qualsiasi del computer. Il nastro inizierà a girare ed il computer caricherà il file.

Lo schermo visualizzerà il seguente messaggio:

```
Loading NOMEFILE block 1
```

...e quindi aggiornerà il numero del blocco man mano che vengono caricati.

Se il primo carattere del nome del file è ! il messaggio precedente non verrà visualizzato e non sarà necessario premere un tasto (occorre ricordarsi però di premere il tasto **PLAY** del registratore). Se i propri programmi usano il ! ma possono operare anche su disco, in fase di lettura del disco il punto esclamativo verrà ignorato. Si noti che il ! NON occupa un carattere nel nome del file su cassetta o su disco.

L'abbandono del comando tramite il tasto **[ESC]** provoca la visualizzazione del seguente messaggio:

```
Broken in
```

Parole chiave associate: CHAIN, CHAIN MERGE, MERGE, RUN, SAVE.

LOCATE

LOCATE [#<canale>,]<coordinata x>,<coordinata y>

```
10 MODE 1
20 FOR n=1 TO 20
30 LOCATE n,n
40 PRINT CHR$(143);"locazione";
50 PRINT n;",";n
60 NEXT
run
```

COMANDO: Localizza il cursore sul testo nel flusso indicato, alla posizione specificata dalle coordinate x e y, dove 1,1 è l'angolo superiore sinistro delcanale (finestra). Il canale #0 è il canale per difetto.

Parole chiave associate: WINDOW.

LOG

LOG(<espressione numerica>)

```
PRINT LOG(9999)
 9.12024037
```

FUNZIONE: Calcola il logaritmo naturale dell'<espressione numerica>, che deve essere maggiore di zero.

Parole chiave associate: EXP, LOG10

LOG10

LOG10(<espressione numerica>)

```
PRINT LOG10(9999)
 3.99995657
```

FUNZIONE: Calcola il logaritmo in base 10 dell'<espressione numerica>, che deve essere maggiore di zero.

Parole chiave associate: EXP, LOG.

LOWER$

LOWER$(<espressione di stringa>)

```
10 a$="GUARDA COME CAMBIANO LE LETTERE IN "
20 PRINT LOWER$(a$+"MINUSCOLO")
run
```

FUNZIONE: Restituisce una nuova espressione di stringa che risulta essere una copia dell'<espressione di stringa> specificata, nella quale tutti i caratteri alfabetici compresi tra A e Z vengono convertiti in minuscolo. La funzione risulta utile per elaborare dati in ingresso che possono essere in caratteri maiuscoli o minuscoli.

Parole chiave associate: UPPER$.

MASK

MASK [<espressione intera>][,<assegnazione primo punto>]

```
10 MODE 0:INK 5,21:INK 8,16
20 MOVE -100*RND,400*RND
30 WHILE XPOS<640
40 FOR x=1 TO 8
50 MASK 2 (8-x)
60 DRAWR 32,0,x,1:MOVER -32,0
70 NEXT
80 MOVER 34,0
90 WEND:GOTO 20
run
```

COMANDO: Predispone la “maschera” che deve essere usata quando vengono tracciate delle linee. Il valore binario dell’<espressione intera> compresa tra 0 e 255 assegna ai bit di ogni gruppo di pixel adiacenti il valore ON (1) o OFF (0).

L’<assegnazione primo punto> determina se il primo punto della linea deve essere inserito (1) o meno (0).

E’ possibile omettere i parametri, ma non entambi contemporaneamente. Se un parametro viene omesso, quel particolare valore non viene modificato.

Parole chiave associate: DRAW, DRAWR, GRAPHICS PAPER, GRAPHICS PEN.

MAX

MAX(<lista di:<espressione numerica>)

```
10  n=66
20  PRINT MAX(1,n,3,6,4,3)
run
 66
```

FUNZIONE: Restituisce il valore massimo della <lista di:<espressioni numeriche>.

Parole chiave associate: MIN.

MEMORY

MEMORY <espressione di indirizzo>

```
MEMORY &20AA
```

COMANDO: Alloca la quantità di memoria BASIC disponibile assegnando l’indirizzo del byte più elevato.

NOTA: BASIC utilizza solo i primi 64K di memoria.

Parole chiave associate: FRE, HIMEM, SYMBOL, SYMBOL AFTER.

MERGE

MERGE <nomefile>

```
MERGE   carica.bas "
```

COMANDO: Carica un programma da disco e lo aggiunge al programma attualmente in memoria.

Si noti che i numeri di linea del vecchio programma esistenti anche nel nuovo programma che deve essere sottoposto a MERGE verranno riscritti dalle linee del nuovo programma.

I file protetti (salvati con ,p) NON possono essere inseriti con MERGE nel programma corrente.

464: Non è necessario specificare il nome del file su cassetta che si vuole caricare.

Verrà indicato:

```
Press PLAY then any key:
```

...a tal punto occorre premere il tasto **PLAY** del registratore e quindi un tasto qualsiasi del computer. Il nastro inizierà a girare ed il computer caricherà il file.

Lo schermo visualizzerà il seguente messaggio:

```
Loading NOMEFILE block 1
```

...e quindi aggiornerà il numero del blocco man mano che vengono caricati.

Se il primo carattere del nome del file è ! il messaggio precedente non verrà visualizzato e non sarà necessario premere un tasto (occorre ricordarsi però di premere il tasto **PLAY** del registratore). Se i propri programmi usano il ! ma possono operare anche su disco, in fase di lettura del disco il punto esclamativo verrà ignorato. Si noti che il ! NON occupa un carattere nel nome del file su cassetta o su disco.

L'abbandono del comando tramite il tasto **[ESC]** provoca la visualizzazione del seguente messaggio:

```
Broken in
```

Parole chiave associate: CHAIN, CHAIN MERGE, LOAD.

MID$

MOD$(<espressione di stringa>,<posizione inizio>[,<lunghezza sottostringa>])

```
10 MODE 1:ZONE 3
20 a$="ENCICLOPEDIA"
30 PRINT "Fammi vedere come e' scritto ";a$
40 PRINT "OK....":PRINT
50 FOR n=1 TO LEN(a$)
60 PRINT MID$(a$,n,1),
70 FOR t=1 TO 700:NEXT t,n
80 PRINT:PRINT
90 INPUT "Adesso inserisci un'altra parola";a$
100 GOTO 50
run
```

FUNZIONE: Restituisce una nuova sottostringa, che inizia alla <posizione inizio> dell'<espressione di stringa>, ed è lunga <lunghezza sottostringa> caratteri. Se il parametro <lunghezza sottostringa> non viene specificato, viene restituito ciò che rimane della <espressione di stringa> dopo la <posizione inizio>.

Se la <posizione inizio> è maggiore della lunghezza totale dell'<espressione di stringa>, viene restituita una stringa vuota. La <posizione inizio> deve essere compresa tra 0 e 255.

Parole chiave associate: LEFT$, RIGHT$.

MID$

MID$(<variabile di stringa>,<posizione inserimento>[,<nuova lunghezza stringa>])=<nuova espressione stringa>

```
10 a$="ciao"
20 MID$(a$,2,2)="XX"
30 PRINT a$
run
cXXo
```

COMANDO: Inserisce la <nuova espressione stringa> nella stringa specificata dalla <variabile di stringa>, iniziando dalla <posizione inserimento> e occupando come numero di caratteri la <nuova lunghezza stringa>.

Si noti che quando MID$ viene usato come comando, dovrà essere utilizzata una <variabile di stringa> come a$, e NON una costante come "ciao".

Parole chiave associate: LEFT$, RIGHT$.

MIN

MIN(<lista di:<espressione numerica>)

```
PRINT MIN(3,6,2.999,8,9,)
 2.999
```

FUNZIONE: Restituisce il valore minimo della <lista di:<espressioni numeriche>.

Parole chiave associate: MAX.

MOD

<argomento> MOD <argomento>

```
PRINT 10 MOD 3
 1
PRINT 10 MOD 5
 0
```

OPERATORE: Retistuisce il resto ottenuto dalla divisione del primo <argomento> per il secondo <argomento> ed eliminando i componenti interi: MODulo.

Parole chiave associate: **nessuna**.

MODE

MODE <espressione intera>

```
10 m=m+1:IF m>2 THEN m=0
20 MODE m
30 PRINT "Questo è mode";m
40 PRINT "ora premi un tasto"
50 IF INKEY$="" THEN 50 ELSE 10
run
```

COMANDO: Modifica il modo schermo (0,1 o 2) e pulisce lo schermo con l'inchiostro 0 (che può non essere il corrente inchiostro della carta). Tutte le finestre di testo e grafica vengono riportate a schermo intero, e i cursori grafico e di testo vengono diretti alle loro rispettive origini.

Parole chiave associate: ORIGIN, WINDOW.

MOVE

MOVE <coordinata x>,<coordinata y>[,[<inchiostro>][,<modo inchiostro>]]

```
10 MODE 1:TAG
20 x=RND*800-100:y=RND*430
30 MOVE x,y
40 PRINT "Sono qui!"
50 GOTO 20
run
```

COMANDO: Sposta il cursore grafico sul punto assoluto specificato dalle <coordinata x> e <coordinata y>. Il parametro <inchiostro> opzionale può essere usato per modificare l'inchiostro della penna grafica, compreso tra i valori 0 e 15.

Il <modo inchiostro> opzionale determina come il successivo inchiostro deve interagire con i caratteri già presenti sullo schermo grafico. I 4 <modi inchiostro> sono:

0: Normale
1: XOR (O esclusivo)
2: AND (E)
3: OR (O)

Parole chiave associate: MOVER, ORIGIN, XPOS, YPOS.

MOVER

MOVER <spostamento x>,<spostamento y>[,[<inchiostro>][,<modo inchiostro>]]

```
10 MODE 1:TAG:MOVE 0,16
20 PRINT "La vita ha i suoi";
30 FOR n=1 TO 10
40 MOVER -32,16
50 PRINT "alti";:NEXT:PRINT " e";
60 FOR n=1 TO 10
70 MOVER -64,-16
80 PRINT "bassi";:NEXT
run
```

COMANDO: Porta il cursore grafico in un punto in posizione relativa alla posizione iniziale. La posizione relativa è specificata da <spostamento x> e <spostamento y>. Il parametro opzionale <inchiostro> può essere utilizzato per modificare l'inchiostro della penna grafica, con valore compreso tra 0 e 15.

Il <modo inchiostro> opzionale determina come il successivo inchiostro deve interagire con i caratteri già presenti sullo schermo grafico. I 4 <modi inchiostro> sono:

0: Normale
1: XOR (O esclusivo)
2: AND (E)
3: OR (O)

Parole chiave associate: MOVE, ORIGIN, XPOS, YPOS.

NEW

NEW

```
NEW
```

COMANDO: Cancella il programma corrente e le variabili presenti in memoria. Le definizioni dei tasti non vengono perse, le caratteristiche di visualizzazione come MODE, PEN, PAPER, INK non vengono modificate e lo schermo non viene pulito.

Parole chiave associate: **nessuna**.

NEXT

NEXT [<lista di:<variabile>]

```
10 FOR a=1 TO 3
20 FOR b=0 TO 26
30 MODE 1
40 PEN a:BORDER b
50 PRINT "penna";a;"cornice";b
60 FOR c=1 TO 500
70 NEXT c,b,a
run
```

COMANDO: Rappresenta il termine di un ciclo FOR. Il comando NEXT può essere anonimo oppure riferirsi al FOR corrispondente. Si noti che nell'esempio precedente la <lista di:<variabile> deve comparire in ordine inverso rispetto ai corrispondenti comandi FOR, in modo che i cicli "annidati" non si sovrappongano.

Parole chiave associate: FOR, STEP, TO.

NOT

NOT <argomento>

```
IF NOT "alan"<"bob" THEN PRINT "corretto" ELSE PRINT "sbagliato"
sbagliato
IF NOT "cane">"gatto" THEN PRINT "corretto" ELSE PRINT "sbagliato"
corretto
....
PRINT NOT 1
 0
PRINT NOT 0
 1
```

OPERATORE: Esegue operazioni booleane bit a bit su interi. Inverte ogni bit nell'argomento.

Ulteriori informazioni relative alla logica si trovano nella parte 2 del capitolo "A vostra disposizione...".

Parole chiave associate: AND, OR, XOR.

ON BREAK CONT

ON BREAK CONT

```
10 ON BREAK CONT
20 PRINT "Il programma CONTinuera' quando si cerchera' di *interrom-
pere* con [ESC]":PRINT
30 FOR t=1 TO 1000:NEXT:GOTO 20
run
```

COMANDO: Disabilita l'azione del tasto **[ESC]**, che non blocca più il programma, facendone invece proseguire l'esecuzione. Si dovrà porre molta attenzione quando si usa questo comando, poichè il programma proseguirà la sua esecuzione fino ad un completo riavvio del computer. Di conseguenza sarà necessario memorizzare un tale programma con SAVE prima di eseguirlo.

ON BREAK CONT può essere disabilitato all'interno di un programma mediante ON BREAK STOP.

Parole chiave associate: ON BREAK GOSUB, ON BREAK STOP.

ON BREAK GOSUB

ON BREAK GOSUB <numero linea>

```
10 ON BREAK GOSUB 40
20 PRINT "programma in esecuzione"
30 GOTO 20
40 CLS:PRINT "hai premuto [ESC] ";
50 PRINT "due volte richiama la routine GOSUB"
60 FOR t=1 TO 2000:NEXT
70 RETURN
run
```

COMANDO: Ordina a BASIC di passare alla sottoroutine specificata nel <numero linea> quando il tasto **[ESC]** viene premuto due volte.

Parole chiave associate: ON BREAK CONT, ON BREAK STOP, RETURN.

ON BREAK STOP

ON BREAK STOP

```
10 ON BREAK GOSUB 40
20 PRINT "programma in esecuzione"
30 GOTO 20
40 CLS:PRINT "hai premuto [ESC] ";
50 PRINT "due volte richiama la routine GOSUB"
60 FOR t=1 TO 2000:NEXT
65 ON BREAK STOP
70 RETURN
run
```

COMANDO: Disabilita i comandi ON BREAK CONT e ON BREAK GOSUB, in modo che successive pressioni del tasto **[ESC]** possano arrestare il programma. Nell'esempio precedente, il comando ON BREAK GOSUB avrà effetto una sola volta, poichè viene disabilitato dalla linea 65 nella sottoroutine ON BREAK.

Parole chiave associate: ON BREAK CONT, ON BREAK GOSUB.

ON ERROR GOTO

ON ERROR GOTO <numero linea>

```
10 ON ERROR GOTO 60
20 CLS:PRINT "Se incontri errori, ";
30 PRINT "lista il programma"
40 FOR t=1 TO 4000:NEXT
50 GOTO 100
60 PRINT "Errore rilevato alla linea";
70 PRINT ERL:PRINT:LIST
run
```

COMANDO: Quando nel programma viene rilevato un errore, si raggiunge il <numero linea> specificato.

La forma del comando ON ERROR GOTO 0 elimina la trappola per gli errori e ripristina il normale trattamento degli errori di BASIC.

Si veda inoltre il comando RESUME.

Parole chiave associate: DERR, ERL, ERR, ERROR, RESUME.

ON <espressione> GOSUB

ON <selettore> GOSUB <lista di:<numero linea>

```
10 PAPER 0:PEN 1:INK 0,1
20 CLS:PRINT "MENU DI OPZIONI":PRINT
30 PRINT "1 - Modifica cornice":PRINT
40 PRINT "2 - Modifica penna":PRINT
50 PRINT "3 - Modifica modo":PRINT
60 INPUT "Inserisci selezione";x
70 ON x GOSUB 90,110,130
80 GOTO 20
90 b=b-1:IF b<0 THEN b=26
100 BORDER b:RETURN
110 p=p-1:IF p<2 THEN p=26
120 INK 1,p:RETURN
130 m=m-1:IF m<0 THEN m=2
140 MODE m:RETURN
run
```

COMANDO: Seleziona una linea di subroutine da raggiungere, in base al valore del <selettore>, che deve essere un'espressione intera positiva compresa tra 0 e 255. L'ordine dei valori del <selettore> determina il <numero linea> che deve essere selezionato dalla <lista di:<numeri linea>. Nell'esempio precedente, selezionando 1 si ottiene un salto alla linea 90, selezionando 2 si ottiene un salto alla linea 110 e selezionando 3 si ottiene un salto alla linea 130.

Se il valore del <selettore> è zero, o maggiore della quantità di <numeri linea> presenti nel comando, non verrà selezionata alcuna sottoroutine.

Parole chiave associate: RETURN.

ON <espressione> GOTO

ON <selettore> GOTO <lista di:<numero linea>

```
10  CLS:PRINT "MENU DI OPZIONI":PRINT
20  PRINT "1 - Lista programma":PRINT
30  PRINT "2 - Modifica e aggiungi":PRINT
40  PRINT "3 - Cataloga disco":PRINT
50  INPUT "Inserisci selezione";x
60  ON n GOTO 80,90,100
70  GOTO 10
80  LIST
90  AUTO
100  CAT
run
```

COMANDO: Seleziona una linea da raggiungere, in base al valore del <selettore>, che deve essere un'espressione intera positiva compresa tra 0 e 255. L'ordine dei valori del <selettore> determina il <numero linea> che deve essere selezionato dalla <lista di:<numeri linea>. Nell'esempio precedente, selezionando 1 si ottiene un salto alla linea 80, selezionando 2 si ottiene un salto alla linea 90 e selezionando 3 si ottiene un salto alla linea 100.

Se il valore del <selettore> è zero, o maggiore della quantità di <numeri linea> presenti nel comando, non verrà selezionata alcuna sottoroutine.

Parole chiave associate: **nessuna**

ON SQ GOSUB

ON SQ(<canale>) GOSUB <numero linea>

```
10 ENV 1,15,-1,1
20 ON SQ(1) GOSUB 60
30 MODE 0:ORIGIN 0,0,200,440,100,300
40 FOR x=1 TO 13:FRAME:MOVE 330,200,x
50  FILL x:NEXT:GOTO 40
60  READ s:IF s=0 THEN RESTORE:GOTO 60
70 SOUND 1,s,25,15,1
80  ON SQ(1) GOSUB 60:RETURN
90 DATA 50,60,90,100,35,200,24,500,0
run
```

COMANDO: Esegue una subroutine BASIC quando è presente un elemento vuoto nella coda di suono data. Il <canale> deve essere un'espressione intera contenente uno dei valori seguenti:

1: per il canale A
2: per il canale B
4: per il canale C

Maggiori informazioni relative ai suoni si trovano nella parte 2 del capitolo A vostra disposizione"

Parole chiave associate: RETURN, SOUND, SQ

OPENIN

OPENIN <nomefile>

```
10 REM OPEN di un file di INgresso da disco
20 OPENIN"filedati":INPUT #9,a,a$
30 CLOSEIN:PRINT "I 2 valori sono:"
40 PRINT:PRINT a,a$
run
```

COMANDO: Apre (OPEN) un file in INput da disco, perchè venga usato nel programma corrente. Il file di input aperto deve essere di tipo ASCII.

L'esempio precedente funzionerà solo dopo aver creato il file mostrato nel prossimo esempio (sotto OPENOUT).

464: Se si desidera che venga caricato il primo file contenuto sul nastro, non è necessario specificare il nome del file.

Verrà visualizzato il messaggio:

```
Press PLAY then any key:
```

...a tal punto occorre premere il tasto **PLAY** del registratore e quindi un tasto qualsiasi del computer. Il nastro inizierà a girare ed il computer inizierà a caricare dal nastro i primi 2 Kbyte e li memorizzerà nel "buffer del file". A questo punto i dati verranno prelevati dal buffer; quando questo sarà vuoto verrà nuovamente visualizzato il messaggio:

```
Press PLAY then any key:
```

... e caricherà da cassetta i successivi 2 Kbyte.

Sullo schermo verrà visualizzato il messaggio:

```
Loading NOMEFILE block 1
```

.. e verrà quindi caricato un altro blocco da 2 Kbyte.

Se il primo carattere del nome del file è ! il messaggio precedente non verrà visualizzato e non sarà necessario premere un tasto (occorre ricordarsi però di premere il tasto **PLAY** del registratore). Se i propri programmi usano il ! ma possono operare anche su disco, in fase di lettura del disco il punto esclamativo verrà ignorato. Si noti che il ! NON occupa un carattere nel nome del file su cassetta o su disco.

L'abbandono del comando tramite il tasto [ESC] provoca la visualizzazione del seguente messaggio:

```
Broken in
```

Parole chiave associate: CLOSEIN, EOF.

OPENOUT

OPENOUT <nomefile>

```
10 REM OPEN di un file di uscita su disco
20 INPUT "Dammi una variabile numerica";a
30 INPUT "Dammi una variabile stringa";a$
40 OPENOUT "filedati"
50 WRITE #9,a,a$
60 CLOSEOUT:PRINT "Dati salvati su disco"
run
```

COMANDO: Apre un file di output su disco.

464: Se si desidera che il file venga salvato senza nome, non è necessario specificare il nome del file.

Innanzitutto, i primi 2 Kbyte del file verranno salvati in un'area di memoria chiamata "buffer del file". All'esaurimento del buffer, verrà visualizzato il messaggio:

```
Press REC and PLAY then any key:
```

...a tal punto occorre premere i tasti **RECORD** e **PLAY** del registratore e quindi un tasto qualsiasi del computer. Il nastro inizierà a girare ed il computer inizierà a salvare il contenuto del "buffer del file". Il computer quindi riempie nuovamente il buffer con i 2 Kbyte successivi e richiederà nuovamente:

```
Press REC and PLAY then any key:
```

... e salverà su cassetta i successivi 2 Kbyte.

Se il buffer non è completamente vuoto ed il programma contiene successivamente l'istruzione CLOSEOUT, il computer salverà su cassetta il contenuto del file presentando il messaggio:

Press REC and PLAY then any key:

Sullo schermo verrà visualizzato il messaggio:

Saving NOMEFILE block

Se il primo carattere del nome del file è ! il messaggio precedente non verrà visualizzato e non sarà necessario premere un tasto (occorre ricordarsi però di premere i tasti **RECORD** e **PLAY** del registratore). Se i propri programmi usano il ! ma possono operare anche su disco, in fase di lettura del disco il punto esclamativo verrà ignorato. Si noti che il ! NON occupa un carattere nel nome del file su cassetta o su disco.

L'abbandono del comando tramite il tasto [ESC] provoca la visualizzazione del seguente messaggio:

Broken in

Parole chiave associate: CLOSEOUT.

OR

<argomento> OR <argomento>

```
IF "alan"<"bob" OR "gatto">"cane" THEN PRINT "corretto" ELSE PRINT "sbagliato"
corretto

IF "bob"<"alan" OR "cane">"gatto" THEN PRINT "corretto" ELSE PRINT "sbagliato"
sbagliato

IF "alan"<"bob" OR "cane">"gatto" THEN PRINT "corretto" ELSE PRINT "sbagliato"
corretto
....
PRINT 1 OR 1
 1
PRINT 0 OR 0
 0
PRINT 1 OR 0
 1
```

OPERATORE: Esegue operazioni booleane bit a bit su interi. Il risultato è 1 a meno che entrambi gli argomenti siano 0.

Ulteriori informazioni relative alla logica si trovano nella parte 2 del capitolo "A vostra disposizione".

Parole chiave associate: AND, NOT, XOR.

ORIGIN

ORIGIN <x>,<y>[,<sinistra>,<destra>,<alto>,<basso>]

```
10 MODE 1:BORDER 13:TAG
20 ORIGIN 0,0,100,540,300,100
30 GRAPHICS PAPER 3:CLG
40 FOR x=550 TO -310 STEP -10
50 MOVE x,206
60 PRINT "Questa e' una finestra grafica ";
70 FRAME:NEXT:GOTO 40
run
```

COMANDO: Assegna ai punti di origine dei grafici 0,0 la posizione specificata dalle coordinate <x> e <y>.

Le dimensioni di una finestra grafica possono inoltre venire definite specificando gli ultimi quattro parametri opzionali. Se le coordinate specificate per la finestra grafica descrivono punti che si trovano oltre il bordo dello schermo, l'estremità della finestra grafica viene assunta al bordo dello schermo.

Parole chiave associate: CLG.

OUT

OUT <numero porta>,<espressione intera>

```
OUT &F8F4,&FF
```

COMANDO: Invia il valore dell'<espressione intera> (compresa tra 0 e 255) all'indirizzo specificato nel <numero porta>.

Non è un comando da usare con leggerezza.

Parole chiave associate: INP, WAIT.

PAPER

PAPER [#<canale>],<inchiostro>

```
10  MODE 0:PEN 0:INK 0,13
20  FOR p=1 TO 15
30  PAPER p:CLS
40  LOCATE 7,12:PRINT "sfondo";p
50  FOR t=1 TO 500;NEXT t,p
run
```

COMANDO: Assegna il colore dello sfondo per i caratteri. Quando i caratteri vengono scritti sullo schermo di testo, la cella del carattere viene riempita con l'<inchiostro> dello sfondo (di valore compreso tra 0 e 15) prima che il carattere venga scritto (se non è stato selezionato il modo trasparente).

Se viene omesso il <canale>, viene considerato per difetto l'inchiostro dello sfondo relativo al canale #0.

Il numero dei diversi colori di sfondo accettati dipende dal modo dello schermo.

Parole chiave associate: GRAPHICS PAPER, INK, PEN.

PEEK

PEEK(<espressione di indirizzo>)

```
10  MODE 1:ZONE 7
20 WINDOW 1,40,1,2:WINDOW #1,1,40,3,25
30  PRINT "Indirizzo memoria"
40  LOCATE 20,1:PRINT "Contenuto memoria"
50  FOR n=0 TO 65535
60 p=PEEK(n)
70 PRINT #1,n,"(&";HEX$(n);")";
80 PRINT #1,TAB(20);p,"(&";HEX$(p);")"
90  NEXT
run
```

FUNZIONE: Restituisce il contenuto della locazione di memoria dello Z80 specificata nell'<espressione di indirizzo>, che deve essere compresa tra &0000 e &FFFF (da 0 a 65535). In ogni caso PEEK restituisce il valore dell'indirizzo di RAM specificato (non di ROM), e sarà compreso tra &00 e &FF (da 0 a 255).

Parole chiave associate: POKE.

PEN

PEN [#<canale>,][<inchiostro>][,<modo sfondo>]

```
10  MODE 0:PAPER 0:INK 0,13
20  FOR p=1 TO 15
30  PEN p:PRINT SPACE$(47);"penna";p
40  FOR t=1 TO 500:NEXT t,p:GOTO 20
run
```

COMANDO: Assegna il valore dell'<inchiostro> (compreso tra 0 e 15) che deve essere usato quando si scrive sul canale di schermo specificato (il canale #0 se non specificato). Il parametro <modo sfondo> può essere trasparente (1) oppure opaco (0).

E' possibile omettere uno degli ultimi due parametri, ma non entrambi contemporaneamente. Se viene omesso un parametro, quel particolare valore non viene modificato.

Parole chiave associate: PAPER.

PI

PI

```
PRINT PI
 3.14159265
```

FUNZIONE: Restituisce il valore del rapporto tra la circonferenza e il suo diametro.

Parole chiave associate: DEG, RAD.

PLOT

PLOT <coordinata x>,<coordinata y>[,[<inchiostro>][,<modo inchiostro>]]

```
10  MODE 1:BORDER 0:PAPER 0:PEN 1
20  INK 0,0:INK 1,26:INK 2,13,26:DEG
30  FOR x=1 TO 360:ORIGIN 320,200
40  DRAW 50*COS(x),50*SIN(x),1
50  PLOT 100*COS(x),25*SIN(x):NEXT
60  ORIGIN 0,0:t=TIME+700:WHILE TIME<t
70  PLOT RND*640,RND*400:WEND
80  PLOT RND*640,RND*400,2
90  GOTO 90
run
```

COMANDO: Disegna un punto sullo schermo grafico, nella posizione assoluta specificata dalle coordinate x e y. L'<inchiostro> con cui disegnare il punto può essere specificato (in valore compreso tra 0 e 15).

Il <modo inchiostro> opzionale determina come l'inchiostro con cui si scrive interagisce con quanto già presente sullo schermo grafico. I quattro <modi inchiostro> sono:

0: Normale
1: XOR (O esclusivo)
2: AND (E)
3: OR (O)

Parole chiave associate: GRAPHICS PEN, PLOTR.

PLOTR

PLOTR <spostamento x>,<spostamento y>[,[<inchiostro>][,<modo inchiostro>]]

```
10 REM Usa i tasti cursore per tracciare linee
20 BORDER 0:GRAPHICS PEN 1
30 MODE 1:PLOT 320,200
40 IF INKEY(0)=0 THEN PLOTR 0,1
50 IF INKEY(1)=0 THEN PLOTR 1,0
60 IF INKEY(2)=0 THEN PLOTR 0,-1
70 IF INKEY(8)=0 THEN PLOTR -1,0
80 IF INKEY(9)=0 THEN 30:REM copia=pulisci
90 GOTO 40
run
```

COMANDO: Disegna un punto sullo schermo grafico, nella posizione specificata da <spostamento x> e <spostamento y>, relativa alla corrente posizione del cursore grafico. L'<inchiostro> in cui disegnare il punto può essere specificato (in valore compreso tra 0 e 15).

Il <modo inchiostro> opzionale determina come l'inchiostro con cui si scrive interagisce con quanto già presente sullo schermo grafico. I quattro <modi inchiostro> sono:

0: Normale
1: XOR (O esclusivo)
2: AND (E)
3: OR (O)

Parole chiave associate: GRAPHICS PEN, PLOT.

POKE

POKE <espressione di indirizzo>,<espressione intera>

```
10 FOR m=49152 TO 65535
20 POKE m,100
30 NEXT
run
```

COMANDO: Scrive il valore dell'<espressione intera> (compresa tra 0 e 255) direttamente nella memoria dello Z80 (RAM) nell'<espressione di indirizzo> specificata.

Non è un comando da usare con leggerezza.

Parole chiave associate: PEEK.

POS

POS(#<canale>)

```
10  MODE 1:BORDER 0:LOCATE 8,2
20  PRINT "usare i tasti cursore destra/sinistra"
30 WINDOW 1,40,12,12:CURSOR 1,1
40  FOR n=1 TO 19:PRINT CHR$(9);:NEXT
50  IF INKEY(1)<>-1 THEN PRINT CHR$(9)
60  IF INKEY(8)<>-1 THEN PRINT CHR$(8)
70 LOCATE #1,2,24
80  PRINT #1,"cursore di testo ";
90  PRINT #1,"posizione orizzontale =";
100  PRINT #1,POS(#0):GOTO 50
run
```

FUNZIONE: Restituisce la corrente POSizione orizzontale del cursore di testo relativa all'estremità sinistra della finestra di testo. Il <canale> DEVE essere specificato, e NON ha valore #0 per difetto.

POS(#8) restituisce la corrente posizione orizzontale del carrello della stampante, in cui 1 è l'estremo sinistro.

POS(#9) restituisce la posizione logica nel canale del disco, cioè il numero di caratteri da stampare inviati alla stampante dopo l'ultimo carattere di ritorno carrello.

Parole chiave associate: VPOS, WINDOW.

PRINT

PRINT[#<canale>,][<lista di:<elementi di stampa>]

```
10 a$="piccola"
20 b$="questa e' una stringa piu' lunga"
30 PRINT a$;a$
40 PRINT a$;a$
50 PRINT
60 PRINT b$;b$
70 PRINT b$;b$
run
```

COMANDO: Stampa la <lista di:<elementi di stampa> sul canale specificato (sul canale #0 se non viene specificato un <canale >).

Si noti che quando viene usato un punto e virgola ; per indicare al computer di stampare l'<elemento di stampa> successivo accanto all'elemento precedente, BASIC controlla innanzitutto se l'<elemento di stampa> può essere inserito sulla stessa linea, altrimenti verrà stampato su una nuova linea, senza tener conto del punto e virgola.

Si noti inoltre che quando viene usata una virgola , per indicare al computer di stampare l'<elemento di stampa> seguente nella successiva zona di stampa, BASIC verifica se l'elemento precedente non ha oltrepassato i limiti della zona corrente, altrimenti il seguente <elemento di stampa> viene stampato in una nuova zona.

PRINT SPC
PRINT TAB

PRINT[#<canale>,][<lista di:<elemento di stampa>][;]
[SPC (<espressione intera>)[<lista di:<elemento di stampa>]

PRINT[#<canale>,][<lista di:<elemento di stampa>][;]
[TAB(<espressione intera>)[<lista di:<elemento di stampa>]

```
10 PRINT "questa e' la funzione spc"
20 FOR x=6 TO 15
30 PRINT SPC(5)"a";SPC(x)"b"
40 NEXT
50 PRINT "questa e' la funzione tab"
60 FOR x=6 TO 15
70 PRINT TAB(5)"a";TAB(x)"b"
80 NEXT
run
```

SPC stampa il numero di spazi identificati nell'<espressione intera> e quindi l'<elemento di stampa> immediatamente accanto agli spazi (assumendo che l'<elemento di stampa> successivo possa essere inserito sulla linea). Di conseguenza non è necessario concludere SPC con un punto e virgola.

TAB stampa il numero di spazi relativi all'estremità sinistra della finestra di testo e quindi l'<elemento di stampa> immediatamente accanto agli spazi (assumendo che l'<elemento di stampa> successivo possa essere inserito sulla linea). Di conseguenza non è necessario concludere TAB con un punto e virgola. Se la posizione corrente è maggiore della posizione richiesta, viene inserito un carattere di ritorno carrello, seguito dagli spazi necessari per raggiungere la posizione richiesta sulla linea successiva.

PRINT USING

PRINT[#<canale>][<lista di:elementi di stampa>][;]
[USING <maschera di formato>][<separatore><espressione>]

```
10 FOR x=1 TO 10
20 n=100000*(RND↑5)
30 PRINT "merci";USING "########,.##";n
40 NEXT
run
```

PRINT USING permette di specificare il formato di stampa dell'espressione restituita dal comando PRINT. Questo avviene specificando una <maschera di formato> cui il risultato stampato deve corrispondere. Il <separatore> è una virgola o un punto e virgola. La <maschera di formato> è una espressione di stringa realizzata mediante i seguenti "indicatori di formato di campo":

Formati numerici

All'interno del numero:

\# Ciascun # specifica una posizione di cifra.
Esempio di maschera: ######

. Specifica la posizione del punto decimale.
Esempio di maschera: ######.##

, (Specifica la posizione di una cifra). Può apparire solo PRIMA del punto decimale. Specifica che le cifre prima del punto decimale devono essere suddivise in gruppi di tre (per le migliaia), separati da virgole.
Esempio di maschera: ######,.##

Intorno al numero:

££ (Specifica la posizione di due cifre). Specifica che un simbolo £ deve essere stampato immediatamente prima della prima cifra o del punto decimale (dopo un qualsiasi segno iniziale). Si noti che £ occuperà una delle posizioni delle cifre.
Esempio di maschera: ££######,.##

** (Specifica la posizione di due cifre). Specifica che tutti gli spazi iniziali devono essere sostituiti con * asterischi.
Esempio di maschera: **######,.##

**£ (Specifica la posizione di tre cifre) Rappresenta la combinazione delle due opzione ££ e ** combinate, cioè la presenza di asterischi * iniziali e di un simbolo £.
Maschera di esempio: **£######,.##

$$ (Specifica la posizione di due cifre). Specifica che un simbolo $ deve essere stampato immediatamente prima della prima cifra o del punto decimale (dopo un qualsiasi segno iniziale). Si noti che $ occuperà una delle posizioni delle cifre.
Maschera di esempio:$$######,.##

**$ (Specifica la posizione di tre cifre). Rappresenta la combinazione delle due opzioni $$ e ** combinate, cioè la presenza di asterischi * iniziali e di un simbolo $.
Maschera di esempio: **$######,.##

\+ Specifica che deve essere stampato + o -, nel modo appropriato. Se all'inizio della maschera compare +, il segno + viene stampato immediatamente prima del numero (e di qualsiasi altro segno iniziale). Se il + appare alla fine della maschera, il segno viene stampato dopo il numero (e qualsiasi eventuale parte esponenziale).
Maschera di esempio: +####.####

\- Il segno - può apparire solo alla FINE di una maschera. Specifica che - deve essere stampato dopo ogni numero negativo (e qualsiasi eventuale parte esponenziale). Se il numero è positivo, verrà stampato uno spazio. Un segno - viene stampato per difetto prima di un numero negativo, se non richiesto diversamente tramite l'uso di questa maschera.
Maschera di esempio: ####.####-

↑↑↑↑ Specifica che il numero deve essere stampato utilizzando l'opzione esponente. I caratteri ↑↑↑↑ nella maschera devono apparire DOPO le posizioni delle cifre, ma PRIMA di segni + o - alla fine della maschera.
Maschera di esempio: #.####↑↑↑↑+

La <maschera di formato> per un numero non può oltrepassare i 20 caratteri di lunghezza. I numeri vengono arrotondati per la quantità di cifre stampate.

Se la maschera di formato è troppo piccola per l'espressione in ingresso, come ad esempio

```
PRINT USING "####";12345678
```

il risultato stampato NON viene ridotto per la maschera, ma viene stampato completamente, e fatto precedere da un segno % che indica un "fallimento di formato".

Formati di stringhe

```
10  CLS:a$="abcdefghijklmnopqrst"
20 PRINT "espressione in ingresso= ";a$
30 PRINT:PRINT "! specificatore= ";
40 PRINT USING "!";a$
50 PRINT :PRINT "\spazi\ specificatore= ";
60  PRINT USING "\    \";a$
70  PRINT:PRINT "& specificatore= ";
80  PRINT USING "&";a$
90  GOTO 90
run
```

! Specifica che solo il primo carattere della stringa deve essere stampato.
Maschera di esempio: !

\<spazi>\
Specifica che solo i primi x caratteri della stringa devono essere stampati, dove x è uguale alla lunghezza della maschera (incluse le barre inverse).
Maschera di esempio: \   \

& Specifica che l'intera stringa deve essere stampata "come è".
Maschera di esempio: &

La <maschera di formato> per una stringa non deve essere maggiore di 255 caratteri.

Le <maschere di formato> numeriche e di stringa possono essere rappresentate da variabili di stringa, come ad esempio:

```
10  a$="££ ######,.##"
20  b$="!"
30  PRINT USING a$;12345.6789;
40  PRINT USING b$;"sterline"
run
```

Ulteriori informazioni relative alla stampa formattata si trovano nella parte 2 del capitolo "A vostra disposizione".

Parole chiave associate: SPC, TAB, USING, ZONE.

RAD

RAD

```
RAD
```

COMANDO: Attiva il modo di calcolo in RADianti. BASIC ritorna al modo radianti quando il computer viene acceso o riavviato, oppure quando si inviano i comandi NEW, CLEAR, LOAD, RUN, ecc.

Parole chiave associate: ATN, COS, DEG, SIN, TAN.

RANDOMIZE

RANDOMIZE[<espressione numerica>]

```
RANDOMIZE 123.456
PRINT RND
 0.258852139
```

COMANDO: Fissa il "seme" per il generatore casuale specificato nell'<espressione numerica>. Il generatore BASIC di numeri casuali produce una sequenza pseudo-casuale nella quale ogni numero dipende dal numero precedente, a partire dal numero seme specificato. La sequenza ottenuta è sempre la stessa. RANDOMIZE assegna il nuovo valore iniziale per il generatore di numeri casuali sia al valore specificato, sia ad un valore introdotto dall'utente se viene omessa l'<espressione numerica>.

RANDOMIZE TIME produce una sequenza difficile da ripetere.

Parole chiave associate: RND.

READ

READ <lista di:<variabile>

```
10  FOR n=1 TO 8
20  READ a$,c
30 PRINT a$;" ";:SOUND 1,c:NEXT
40 DATA queste,478,sono,426,8,379,note
50 DATA 358,di,319,una,284,scala,253,musicale,239
run
```

COMANDO: Legge dati dalle istruzioni DATA e li assegna a variabili, spostando quindi automaticamente il puntatore al successivo elemento dell'istruzione DATA. Il comando RESTORE può essere utilizzato per riportare il puntatore all'inizio di una istruzione DATA.

Ulteriori informazioni relative ai dati si trovano nella parte 2 del capitolo "A vostra disposizione".

Parole chiave associate: DATA, RESTORE.

RELEASE

RELEASE <canali di suono>

```
10 SOUND 65,1000,100
20  PRINT "Premi R per attivare il suono"
30 IF INKEY(50)=-1 THEN 30
40 RELEASE 1
run
```

COMANDO: Rilascia i canali di suono che si trovano in uno stato di "blocco" nel comando SOUND.

I parametri <canali di suono> devono presentare un valore intero compreso tra 1 e 7, che opera nel modo seguente:

1: Rilascia il canale A
2: Rilascia il canale B
3: Rilascia i canali A e B
4: Rilascia il canale C
5: Rilascia i canali A e C
6: Rilascia i canali B e C
7: Rilascia i canali A e B e C

Ulteriori informazioni relative ai suoni si trovano nella parte 2 del capitolo "A vostra disposizione".

Parole chiave associate: SOUND.

REM

REM <resto della linea>

```
10  REM Intergalactic Hyperspace Mega-Monster
       Invaders Deathchase by AMSOFT
20  REM Copyright AMSOFT 1985
```

COMANDO: Inserisce un commento in un programma. Il resto della linea viene ignorato da BASIC e può contenere qualsiasi carattere, compresi i due punti : che generalmente separano le istruzioni.

Al posto di :REM può essere usato un carattere di apice singolo ' MA NON all'interno di istruzioni DATA, in cui ' viene trattato come parte di una stringa senza apici.

Parole chiave associate: **nessuna**.

REMAIN

REMAIN(<numero timer>)

```
10 AFTER 500,1 GOSUB 40
20 AFTER 100,2 GOSUB 50
30 PRINT "programma in esecuzione":GOTO 30
40  REM Questa ruotine GOSUB non verra' chiamata
perche' disabilitata in linea 80.
50 PRINT:PRINT "Il timer 1 ora sara' ";
60 PRINT "disabilitato da REMAIN."
70  PRINT "Le unita' di tempo restanti erano:";
80 PRINT REMAIN(1)
run
```

FUNZIONE: Restituisce il tempo residuo del timer specificato in <numero timer> (valore compreso tra 0 e 3) e lo disabilita.

Ulteriori informazioni relative alle interruzioni si trovano nella parte 2 del capitolo "A vostra disposizione".

Parole chiave associate: AFTER, DI, EI, EVERY.

RENUM

RENUM[<nuovo numero linea>][,[<vecchio numero linea>][,<incremento>]]

```
10  CLS
20  REM questa diventera' la linea 123
30  REM questa diventera' la linea 124
40  REM questa diventera' la linea 125
RENUM 123,20,1
LIST
```

COMANDO: Rinumera le linee di un programma.

Il parametro <vecchio numero linea> specifica il numero di linea corrente esistente in cui deve iniziare la rinumerazione. Se questo parametro viene omesso, la rinumerazione inizia dalla prima linea del programma.

Il parametro <nuovo numero linea> specifica il nuovo numero di linea per le linee da rinumerare. Se questo parametro viene omesso, il programma verrà rinumerato a partire da linea 10.

Il parametro <incremento> specifica il passo numerico tra ciascuna linea rinumerata. Se viene omesso, il valore del passo sarà 10.

RENUM prende in considerazione tutte le chiamate di linea GOSUB, GOTO e altre. Tuttavia, i riferimenti ai numeri di linea all'interno delle espressioni di stringa, come quelli presenti nei comandi KEY, non vengono alterati, così come i riferimenti a linee all'interno delle istruzioni REM, o le <espressioni di numero di linea> nei comandi CHAIN o CHAIN MERGE.

I numeri di linea sono validi se compresi tra 1 e 65535.

Parole chiave associate: DELETE, LIST.

RESTORE

RESTORE[<numero linea>]

```
10 READ a$:PRINT a$;" ";
20 RESTORE 50
30 FOR t=1 TO 500:NEXT:GOTO 10
40 DATA i dati ripristinati possono essere letti
50 DATA nuovamente
run
```

COMANDO: Ripristina la posizione del "puntatore" all'inizio dell'istruzione DATA specificata nel <numero linea> opzionale. Se il parametro viene omesso, il puntatore viene ripristinato all'inizio della prima istruzione DATA.

Ulteriori informazioni relative ai dati si trovano nella parte 2 del capitolo "A vostra disposizione".

Parole chiave associate: DATA, READ.

RESUME

RESUME[<numero linea>]

```
10  ON ERROR GOTO 60
20  FOR x=10 TO 0 STEP-1:PRINT 1/x:NEXT
30  END
40  PRINT "viene qui dopo l'errore"
50  END
60  PRINT "errore n.";ERR;"nella linea";ERL
70  RESUME 40
run
```

COMANDO: Riprende la normale esecuzione di un programma dopo la rilevazione di un errore trattato mediante un comando ON ERROR GOTO. Se il numero di linea da cui riprendere non è specificato, il programma riprende l'esecuzione dalla stessa linea in cui è stato inizialmente rilevato l'errore. Si elimini il parametro <numero linea> nell'esempio precedente, quindi si esegua nuovamente.

```
70  RESUME
run
```

Parole chiave associate: DERR, ERL, ERR, ERROR, ON ERROR GOTO, RESUME NEXT.

RESUME NEXT

RESUME NEXT

```
10 ON ERROR GOTO 90
20  PRINT "premere [RETURN] ogni volta"
30 INPUT "1";a
40 INPUT "2";a
50 inpot "3";a :REM errore di sintassi!
60 INPUT "4";a
70 INPUT "5";a
80 END
90 PRINT "errore n.";ERR;"nella linea";ERL
100  RESUME NEXT
run
```

COMANDO: Riprende la normale esecuzione di un programma dopo la rilevazione di un errore trattato mediante un comando ON ERROR GOTO.

RESUME NEXT riprende l'esecuzione dalla linea successiva a quella in cui è stato rilevato l'errore.

Parole chiave associate: DERR, ERL, ERR, ERROR, ON ERROR GOTO, RESUME.

RETURN

RETURN

```
10  GOSUB 50:PRINT "dopo il gosub":END
50  FOR n=1 TO 20
60  PRINT "subroutine"
70  NEXT:PRINT
80  RETURN
run
```

COMANDO: Segnala la fine di una subroutine. BASIC rientra dalla subroutine all'istruzione immediatamente successiva al comando GOSUB che l'ha richiamata.

Parole chiave associate: GOSUB

RIGHT$

RIGHT$(<espressione di stringa>,<lunghezza richiesta>)

```
10  MODE 1:a$="CPC6128 computer"
20 FOR n=1 TO 16:LOCATE 41-n,n
30 PRINT RIGHT$(a$,n)
40  NEXT
run
```

FUNZIONE: Restituisce il numero di caratteri (compresi tra 0 e 255) specificati nel parametro <lunghezza richiesta>, dopo averli estratti dalla destra (RIGHT) dell'<espressione di stringa>. Se l'<espressione di stringa> è più breve rispetto alla <lunghezza richiesta>, viene restituita l'intera <espressione di stringa>.

Parole chiave associate: LEFT$, MID$.

RND

RND[(<espressione numerica>)]

```
10 RANDOMIZE
20 FOR x=1 TO -1 STEP -1
30 PRINT "parametro rnd=";x
40 FOR n=1 TO 6
50 PRINT RND(x)
60 NEXT n,x
run
```

FUNZIONE: Restituisce il successivo numero casuale in sequenza se l'<espressione numerica> ha valore positivo oppure non è specificata.

Se l'<espressione numerica> presenta valore zero, RND restituisce una copia dell'ultimo numero casuale generato.

Se l'<espressione numerica> presenta valore negativo, viene iniziata una nuova sequenza di numeri casuali e viene restituito il primo numero di questa sequenza.

Parole chiave associate: RANDOMIZE.

ROUND

ROUND (<espressione numerica>[,<decimali>])

```
10 FOR n=4 TO -4 STEP-1
20 PRINT ROUND (1234.5678,n),
30 PRINT "con espressione intera";n
40 NEXT
run
```

FUNZIONE: Arrotonda l'<espressione numerica> ad un numero con posizioni decimali o potenze di dieci specificati nel parametro <decimali>. Se <decimali> è minore di zero, l'<espressione numerica> viene arrotondata per ottenere un intero assoluto con un numero di <decimali> uguale a zero prima del punto decimale.

Parole chiave associate: ABS, CINT, FIX, INT.

RUN

RUN <espressione di stringa>

```
RUN "disco"
```

COMANDO: Carica un programma BASIC o binario da disco e ne inizia l'esecuzione. Tutti i programmi BASIC precedentemente caricati vengono eliminati dalla memoria.

In questo modo i programmi BASIC protetti possono essere eseguiti direttamente.

Parole chiave associate: LOAD.

RUN

RUN [<numero linea>]

```
RUN 200
```

COMANDO: Inizia l'esecuzione del corrente programma BASIC, dal parametro <numero linea> specificato o dall'inizio del programma se il parametro viene omesso. RUN assegna a tutte le variabili del programma corrente il valore zero o nullo.

I programmi protetti NON possono essere eseguiti in questo modo dopo il caricamento.

Parole chiave associate: CONT, END, STOP.

SAVE

SAVE <nomefile>[,<tipo file>][,<parametri binari>]

SAVE "filedisc.xyz"

memorizza il file in normale modo BASIC non protetto.

SAVE "filedisc.xyz",P

memorizza il file in modo BASIC Protetto.

SAVE "filedisc.xyz",A

memorizza il file in modo ASCII.

SAVE "filedisc.xyz",B,8000,3000,8001

memorizza il file in modo Binario. In questo esempio viene salvata l'area di memoria del computer che inizia all'indirizzo 8000; la lunghezza del file è di 3000 byte e l'indirizzo del punto di ingresso opzionale è 8001.

COMANDO: Memorizza il programma attualmente in memoria su disco. Un file Binario è un'area di memoria salvata su disco. I parametri Binari sono:

<indirizzo iniziale>,<lunghezza file>[,<punto di ingresso>]

La memoria dello schermo può essere salvata come un file Binario: questa operazione è nota come "copia dello schermo" e può essere eseguita mediante il comando:

SAVE "schermo",B,&C000,&4000

Quindi per riportarlo sullo schermo è sufficiente il comando:

LOAD "schermo"

464: Se si desidera che il file venga salvato senza nome, non è necessario specificare il nome del file:

SAVE ""

Verrà visualizzato il messaggio:

Press REC and PLAY then any key:

...a tal punto occorre premere i tasti **RECORD** e **PLAY** del registratore e quindi un tasto qualsiasi del computer. Il nastro inizierà a girare ed il computer salverà il programma su cassetta.

Sullo schermo verrà visualizzato il messaggio:

Saving NOMEFILE block 1

Se il primo carattere del nome del file è ! il messaggio precedente non verrà visualizzato e non sarà necessario premere un tasto (occorre ricordarsi però di premere i tasti **RECORD** e **PLAY** del registratore). Se i propri programmi usano il ! ma possono operare anche su disco, in fase di lettura del disco il punto esclamativo verrà ignorato. Si noti che il ! NON occupa un carattere nel nome del file su cassetta o su disco.

L'abbandono del comando tramite il tasto [ESC] provoca la visualizzazione del seguente messaggio:

Broken in

Parole chiave associate: CHAIN, CHAIN MERGE, LOAD, MERGE, RUN.

SGN

SGN(<espressione numerica>)

```
10  FOR n=200 TO -200 STEP-20
20  PRINT "SGN restituisce";
30  PRINT SGN(n);"per un valore di";n
40  NEXT
run
```

FUNZIONE: Determina il SeGNo dell'<espressione numerica>. SGN restituisce -1 se l'<espressione numerica> è minore di zero, 0 se l'<espressione numerica> è uguale a zero e 1 se l'<espressione numerica> è maggiore di zero.

Parole chiave associate: ABS.

SIN

SIN(<espressione numerica>)

```
10 CLS:DEG:ORIGIN 0,200
20  FOR n=0 TO 720
30  y=SIN(n)
40    PLOT n*640/720,198*y:NEXT
50  GOTO 50
run
```

FUNZIONE: Calcola il seno dell'<espressione numerica>.

Si noti che si possono utilizzare DEG e RAD per ottenere il risultato dell'operazione rispettivamente in gradi o in radianti.

Parole chiave associate: ATN, COS, DEG, RAD, TAN.

SOUND

SOUND <stato canale>,<periodo di tono>[,<durata>[,<volume>[,<inviluppo volume>[,<inviluppo tono>[,<periodo rumore>]]]]]

```
10 FOR z=0 TO 4095
20 SOUND 1,z,1,12
30 NEXT
run
```

COMANDO: Programma un suono. Il comando accetta i seguenti parametri:

Parametro 1: <stato canale>

Il parametro <stato canale> deve contenere un intero compreso tra 0 e 255. Il parametro presenta bit significativi, ed ogni bit del valore binario dello <stato canale> assume il seguente significato:

Bit 0:(decimale 1) invia il suono al canale A (bit meno significativo)
Bit 1:(decimale 2) invia il suono al canale B
Bit 2:(decimale 4) invia il suono al canale C
Bit 3:(decimale 8) rendezvous con canale A
Bit 4:(decimale 16) rendezvous con canale B
Bit 5:(decimale 32) rendezvous con canale C
Bit 6:(decimale 64) blocca il canale di suono
Bit 7:(decimale 128) sblocca il canale di suono (bit più significativo)

Di conseguenza un parametro <stato canale> di valore 68, ad esempio, significa:

Invia a canale C (4) con stato di blocco (64).

Parametro 2: <periodo tono>

Questo parametro definisce il picco del suono, o, in altri termini, "quale nota è" (ad esempio, do re mi fa sol). Ogni nota possiede un particolare numero, che rappresenta il <periodo tono>. Si consulti il capitolo "Riferimenti".

Parametro 3: <durata>

Questo parametro stabilisce la lunghezza del suono, cioè "quanto dura": opera in unità di 0.01 secondi (centesimi di secondo), e se non viene specificato il computer utilizzerà un valore per difetto di 20 (un quinto di secondo).

Se il parametro <durata> è zero, il suono durerà fino al termine dell'inviluppo di volume specificato.

Se il parametro <durata> è negativo, l'inviluppo di volume specificato deve essere ripetuto ABS(<durata>) volte.

Parametro 4: <volume>

Questo parametro specifica il volume iniziale di una nota. Il valore deve essere compreso tra 0 e 15: quando è 0, il volume è disabilitato, mentre a 15 è massimo. Se non viene specificato alcun numero, il computer utilizzerà un valore per difetto di 12.

Parametro 5: <inviluppo volume>

Perchè il volume possa variare durante l'emissione di una nota, è possibile specificare un inviluppo di volume utilizzando il comando separato ENV. Infatti si possono creare fino a 15 inviluppi di volume diversi, di valore compreso tra 1 e 15. Il parametro <inviluppo volume> richiama il numero di riferimento dell'inviluppo di volume necessario al comando SOUND.

Si consulti la descrizione del comando ENV.

Parametro 6: <inviluppo tono>

Perchè il tono o il picco possa variare durante l'emissione di una nota, è possibile specificare un inviluppo di tono utilizzando il comando separato ENT. Infatti si possono creare fino a 15 inviluppi di tono diversi, di valore compreso tra 1 e 15. Il parametro <inviluppo tono> richiama il numero di riferimento dell'inviluppo di tono necessario al comando SOUND. Se nel comando ENT è stato specificato un numero di inviluppo negativo, si usi il valore assoluto di questo numero (cioè senza il segno di negazione) in questo parametro <inviluppo tono> del comando SOUND.

Si consulti la descrizione del comando ENT.

Parametro 7: <rumore>

E' disponibile un insieme di rumore bianco, che può essere disabilitato o aggiunto al suono variando il parametro <suono> tra 0 e 31.

Ulteriori informazioni relative al suono si trovano nella parte 2 del capitolo "A vostra disposizione".

Parole chiave associate: ENT, ENV, ON SQ GOSUB, RELEASE, SQ.

SPACE$

SPACE$(<espressione intera>)

```
10  MODE 1
20  PRINT "Mette 9 spazi tra me";
30 PRINT SPACE$(9);
40 PRINT "e te!"
run
```

FUNZIONE: Crea una stringa di spazi della lunghezza data (compresa tra 0 e 255), specificata nell'<espressione intera>.

Parole chiave associate: SPC, STRING$, TAB.

SPC

(Si veda PRINT SPC).

SPEEK INK

SPEED INK <periodo 1>,<periodo 2>

```
10  BORDER 7,18
20  FOR i=30 TO 1 STEP-1
30  SPEED INK i,i
40  FOR t=1 TO 700:NEXT t,i
run
```

COMANDO: Determina la frequenza con cui si devono alternare due colori di inchiostro specificati in un comando INK o BORDER. <periodo 1> specifica il tempo, in unità di 0.02 secondi (cinquantesimi di secondo) per il primo colore da usare; <periodo 2> specifica il tempo per il secondo colore.

E' necessario usare un po' di buon senso per evitare effetti ipnotici quando si scelgono i colori e le frequenze di ripetizione!

Parole chiave associate: BORDER, INK.

SPEED KEY

SPEED KEY <ritardo inizio>,<periodo ripetizione>

```
10 CLS:FOR k=7 TO 1 STEP-2
20 PRINT "inserisci il tuo nome, poi [RETURN]"
30 SPEED KEY k,k
40 LINE INPUT a$:NEXT
50 PRINT "Che nome buffo!"
run
```

COMANDO: Assegna la frequenza di autoripetizione alla tastiera. Il parametro <ritardo inizio> specifica il tempo, in unità di 0.02 secondi (cinquantesimi di secondo) prima dell'inizio dell'autoripetizione. Il parametro <periodo ripetizione> assegna l'intervallo tra ciascuna autoripetizione di un tasto.

SPEED KEY avrà effetto solo sui tasti dotati per difetto di autoripetizione, oppure definiti con autoripetizione mediante il comando KEY DEF.

Quando si desidera utilizzare piccoli valori di <ritardo inizio>, è consigliabile preprogrammare uno dei tasti numerici per riportare la tastiera al valore di SPEED KEY per difetto di 30,2. Questo si può ottenere con il comando:

```
KEY 0,"SPEED KEY 30,2"+CHR$(13)
```

che riporta SPEED KEY al suo valore per difetto quando viene premuto il tasto 0 del tastierino numerico.

Parole chiave associate: KEY DEF.

SPEED WRITE

SPEED WRITE <espressione intera>

```
SPEED WRITE 1
```

COMANDO: Determina la velocità di memorizzazione dei dati su cassetta (se l'unità a cassette è connessa). Si può scrivere su cassetta alla velocità di 2000 baud (bit per secondo) se l'<espressione intera> è 1, oppure alla velocità per difetto di 1000 baud se l'<espressione intera> è 0. Quando si carica in memoria un file su nastro, il computer seleziona automaticamente la corretta velocità di trasferimento.

Per l'affidabilità di dati importanti, è consigliabile l'uso di SPEED WRITE 0 (valore per difetto).

Il comando SPEED WRITE non ha effetti sulle operazioni con i dischi.

Parole chiave associate: OPENOUT, SAVE.

SQ

SQ(<canale>)

```
10  SOUND 65,100,100
20 PRINT SQ(1)
run
 67
```

FUNZIONE: Restituisce lo stato della coda di suono per il <canale> specificato, che deve essere un'espressione intera, con uno dei seguenti valori:

1: per il canale A
2: per il canale B
4: per il canale C

La funzione SQ restituisce un intero relativo ai bit significativi, comprendente la seguente configurazione di bit:

Bit 0, 1 e 2: numero di elementi liberi nella coda
Bit 3, 4 e 5: stato di rendezvous all'inizio della coda
Bit 6 : l'inizio della coda è bloccato
Bit 7 : il canale è correntemente attivo

dove il bit 0 è il bit meno significativo, mentre il bit 7 è quello più significativo.

Si può vedere inotre che se il bit 6 è attivo, il bit 7 non può essere attivo e viceversa. Analogamente se i bit 3, 4 o 5 sono attivi, i bit 6 e 7 non possono essere attivi.

Ulteriori informazioni relative al suono si trovano nella parte 2 del capitolo "A vostra disposizione"

Parole chiave associate: ON SQ GOSUB, SOUND.

SQR

SQR (<espressione numerica>)

```
PRINT SQR(9)
 3
```

FUNZIONE: Restituisce la radice quadrata dell'<espressione numerica> specificata.

Parole chiave associate: **nessuna**.

STEP

(Si consulti FOR).

STOP

STOP

```
10 FOR n=1 TO 30:PRINT n:NEXT
20 STOP
30 FOR n=31 TO 60:PRINT n:NEXT
run
cont
```

COMANDO: Blocca l'esecuzione di un programma, ma lascia BASIC in uno stato in cui il programma può essere riattivato mediante il comando CONT. STOP può essere utilizzato per interrompere il programma in punti particolari durante un'operazione di debugging.

Parole chiave associate: CONT, END.

STR$

STR$(<espressione numerica>)

```
10 a=&FF :REM 255 esadecimale
20 b=&X1111 :REM 15 binario
30 c$="***"
40 PRINT c$+STR$(a+b)+c$
run
*** 270 ***
```

FUNZIONE: Converte l'<espressione numerica> in una rappresentazione decimale di stringa.

Parole chiave associate: BIN$, DEC$, HEX$, VAL.

STRING$

STRING$(<lunghezza>,<indicatore di carattere>)

```
PRINT STRING$(40,"*")
****************************************
```

FUNZIONE: Restituisce una espressione di stringa formata dal carattere specificato ripetuto il numero di volte (compreso tra 0 e 255) specificato in <lunghezza>. Si noti che l'esempio precedente avrebbe anche potuto essere scritto come:

```
PRINT STRING$(40,42)
****************************************
```

dove l' <indicatore di carattere> 42 si riferisce al valore ASCII del carattere *, risultando così equivalente a PRINT STRING$(40,CHR$(42)).

Parole chiave associate: SPACE$.

SWAP

(Si consulti WINDOW SWAP).

SYMBOL

SYMBOL <numero carattere>,<lista di:<riga>

```
10  MODE 1:SYMBOL AFTER 105
20  riga1=255:REM 11111111 binario
30  riga2=129:REM 10000001 binario
40  riga3=189:REM 10111101 binario
50  riga4=153:REM 10011001 binario
60  riga5=153:REM 10011001 binario
70  riga6=189:REM 10111101 binario
80  riga7=129:REM 10000001 binario
90  riga8=255:REM 11111111 binario
100  PRINT "La linea 110 ridefinisce la lettera i (105).
Batti alcune i e vedrai! Quindi lista il programma."
110  SYMBOL 105,riga1,riga2,riga3,riga4,riga5,riga6,riga7,riga8
run
```

COMANDO: Ridefinisce la forma di un carattere sullo schermo. Ciascun parametro deve contenere un intero compreso tra 0 e 255.

Per allocare spazio nella memoria del 464/6128 per un carattere definito mediante questo comando, è necessario innanzitutto predisporre il computer con:

SYMBOL AFTER x

dove x è uguale o minore del numero del carattere che si desidera ridefinire.

Viene quindi inviato il comando SYMBOL, seguito innanzitutto dal numero di carattere x.

Anche se il valore di x specifica un carattere non direttamente presente su tastiera, il carattere ridefinito può essere stampato sullo schermo mediante il comando:

PRINT CHR$(x)

Dopo SYMBOL x sono presenti 8 parametri che specificano le 8 singole righe componenti il carattere, a partire dalla riga più in alto. Ciascun parametro deve essere compreso tra 0 e 255. La rappresentazione binaria di ciascuno degli 8 parametri determina lo schema di quella particolare riga nel carattere finito.

Se, ad esempio, il primo degli 8 parametri è 1, la riga in alto del carattere avrà rappresentazione binaria 00000001. Dove compare 1, la sezione del carattere viene stampata nel colore della PENna; dove compare 0, la sezione del carattere non è visibile perchè viene stampata nel colore della carta. Di conseguenza la prima riga di questo nuovo carattere avrà un punto nell'angolo superiore destro. Proseguendo questo esempio, si specificheranno gli altri 7 parametri come 3,7,15,31,63,0,0; la loro rappresentazione binaria completa sarà:

parametro(riga)1: 00000001 binario: (decimale 1)
parametro(riga)2: 00000011 binario: (decimale 3)
parametro(riga)3: 00000111 binario: (decimale 7)
parametro(riga)4: 00001111 binario: (decimale 15)
parametro(riga)5: 00011111 binario: (decimale 31)
parametro(riga)6: 00111111 binario: (decimale 63)
parametro(riga)7: 00000000 binario: (decimale 0)
parametro(riga)8: 00000000 binario: (decimale 0)

Osservando la rappresentazione binaria dei precedenti 8 parametri, dovrebbe essere possibile vedere la forma del nuovo carattere. Si assegnino questi parametri al carattere numero 255 mediante il comando:

SYMBOL 255,1,3,7,15,31,63,0,0

Si noti che il valore 0 relativo agli ultimi due parametri significa che è sufficiente digitare:

```
SYMBOL 255,1,3,7,15,31,63
```

Si noti inoltre che è possibile introdurre i parametri in binario, per evitare di dover convertire la "rappresentazione" del simbolo creato in forma decimale (si ricordi di usare il prefisso &X). Ad esempio:

```
SYMBOL255,&X00000001,&X00000011,&X00000111,
&X00001111,&X00011111,&X00111111
```

Per visualizzare il carattere:

```
PRINT CHR$(255)
```

Assegnando i parametri precedenti ad un carattere di tastiera, si otterrebbe il nuovo carattere ad ogni pressione del tasto relativo, oppure in ogni punto in cui avrebbe dovuto comparire il carattere precedente. Inoltre, BASIC non rifiuterebbe il nuovo carattere come incomprensibile, ma lo considererebbe come equivalente al precedente carattere.

Ulteriori informazioni relative ai caratteri definiti da utente si trovano nella parte 2 del capitolo "A vostra disposizione".

Parole chiave associate: SYMBOL AFTER.

SYMBOL AFTER

SYMBOL AFTER <espressione intera>

```
10 CLS
20 SYMBOL AFTER 115
30 PRINT "La linea 40 ridefinisce la s ";
40 SYMBOL 115,0,56,64,64,48,8,8,112
50 PRINT "in S"
60 PRINT "Cancella questa definizione di s"
70 PRINT "digitando: SYMBOL AFTER 240"
run
```

COMANDO: Assegna il numero di caratteri permessi definiti da utente (compresi tra 0 e 256). Il valore per difetto è 240, e consente 16 caratteri definiti da utente (da 240 a 255). Se l'<espressione intera> è 32, tutti i caratteri da 32 a 255 diventano ridefinibili. SYMBOL AFTER 256 impedisce la ridefinizione di caratteri.

Quando viene eseguito un comando SYMBOL AFTER, tutti i caratteri definiti da utente vengono riportati alle loro condizioni per difetto.

SYMBOL AFTER NON avrà effetto se richiamato DOPO l'alterazione di HIMEM mediante il comando MEMORY, oppure mediante l'apertura di un file buffer tramite OPENIN o OPENOUT. In tali circostanze verrà restituito un errore(5) "Improper argument" (a meno che lo stato precedente fosse SYMBOL AFTER 256).

Ulteriori informazioni relative ai caratteri definiti da utente si trovano nella parte 2 del capitolo "A vostra disposizione".

Parole chiave associate: HIMEM, MEMORY, SYMBOL.

TAB

(Si consulti PRINT TAB).

TAG

TAG[#<canale>]

```
10  INPUT "inserisci il tuo nome";a$:CLS
20  PRINT "Sicuramente ti muoverai ";a$
30 TAG
40 x=LEN(a$)*17:y=50+RND*300:MOVE -x,y
50 FOR f=-x TO 640 STEP RND*7+3
60 MOVE f,y:PRINT " ";a$;:FRAME:NEXT
70 FOR b=640 TO -x STEP-RND*7+3
80 MOVE b,y:PRINT a$;" ";:FRAME:NEXT
90 GOTO 40
run
```

COMANDO: Invia qualsiasi testo specificato nel <canale> alla posizione del cursore grafico: in questo modo testo e simboli possono essere inseriti all'interno di grafici, o spostati pixel per pixel invece di carattere per carattere. Se viene omesso il <canale>, viene assunto per difetto il canale #0.

L'angolo superiore sinistro del carattere viene accostato al cursore grafico, mentre i caratteri di controllo non stampabili (come line feed e ritorno di carrello) verranno visualizzati se l'istruzione PRINT non è chiusa da un punto e virgola.

Nel canale per difetto (#0) BASIC disabilita TAG quando ritorna in modo diretto.

Parole chiave associate: TAGOFF.

TAGOFF

TAGOFF[#<canale>]

```
10 MODE 2:TAG :REM Testo e grafica attivato
20 anno=1984:FOR x=1 TO 640 STEP 70
30 MOVE x,400:DRAWR 0,-350
40 anno=anno+1:PRINT anno;:NEXT
50 TAGOFF :REM Testo e grafica DISATTIVATO
60 LOCATE 34,25:PRINT "Cifre annuali"
70 GOTO 70
run
```

COMANDO: Disabilita TAG (testo e grafica) per il <canale> specificato (canale #0 se non indicato) e ridirige il testo alla posizione precedente del cursore di testo usato prima del comando TAG.

Parole chiave associate: TAG.

TAN

TAN(<espressione numerica>)

```
PRINT TAN(45)
 1.61977519
```

FUNZIONE: Calcola la TANgente dell'<espressione numerica>, che deve essere compresa tra -200000 e +200000.

Si noti che si possono utilizzare DEG e RAD per ottenere il risultato dell'operazione rispettivamente in gradi o in radianti.

Parole chiave associate: ATN, COS, DEG, RAD, SIN.

TEST

TEST(<coordinata x>,<coordinata y>)

```
10 CLS
20 PRINT "Stai usando il numero di penna";
30 PRINT TEST(10,386)
40 PRINT "Prova a cambiare PEN e MODE";
50 PRINT "....quindi riesegui il programma."
run
```

FUNZIONE: Porta il cursore grafico alla posizione assoluta specificata dalle coordinate x e y e restituisce il valore dell'inchiostro alla nuova locazione.

Parole chiave associate: MOVE, MOVER, TESTR, XPOS, YPOS.

TESTR

TESTR(<spostamento x>,<spostamento y>)

```
10 MODE 0:FOR x=1 TO 15:LOCATE 1,x
20 PEN x:PRINT STRING$(10,143);:NEXT
30 MOVE 200,400:PEN 1
40 FOR n=1 TO 23:LOCATE 12,n
50 PRINT "penna";TESTR(0,-16):NEXT
run
```

FUNZIONE: Porta il cursore grafico alla posizione specificata dagli spostamenti x e y relativi alla sua posizione corrente e restituisce il valore dell'inchiostro alla nuova locazione.

Parole chiave associate: MOVE, MOVER, TEST, XPOS, YPOS.

THEN

(Si veda IF).

TIME

TIME

```
10 CLS:REM orologio
20 INPUT "ora";ora
30 INPUT "minuto";minuto
40 INPUT "secondo";secondo
50 CLS:dato=INT(TIME/300)
60 WHILE ora<13
70 WHILE minuto<60
80 WHILE tic<60
90 tic=(INT(TIME/300)-dato)+secondo
100 LOCATE 1,1
110 PRINT USING "## ";ora,minuto,tic
120 WEND
130 tic=0:secondo=0:minuto=minuto+1
140 GOTO 50
150 WEND
160 minuto=0:ora=ora+1
170 WEND
180 ora=1
190 GOTO 60
run
```

FUNZIONE: Restituisce il tempo trascorso dall'ultima accensione o dall'ultimo riavvio del computer (ad esclusione dei momenti di lettura o scrittura su disco).

Ogni secondo del tempo reale è uguale al valore restituito:TIME/300.

Parole chiave associate: AFTER, EVERY, WEND, WHILE.

TO

(Si veda FOR).

TROFF
TRON

TROFF
TRON

```
10 TROFF:PRINT:PRINT "TRacciamento OFF"
20  FOR n=1 TO 8
30  PRINT "programma in esecuzione":NEXT
40  IF f=1 THEN END
50 TRON:PRINT:PRINT "TRacciamento ON
60  f=1:GOTO 20
run
```

COMANDO: Esegue un tracciamento dell'esecuzione di un programma stampando ogni numero di linea prima di eseguirlo. Il numero di linea compare tra parentesi quadre [].

TRON attiva il tracciamento, mentre TROFF lo disattiva.

Questa funzione risulta particolarmente utile per lo studio delle sequenze di esecuzione delle linee di un programma immediatamente prima del verificarsi di un errore.

Parole chiave associate: **nessuna**.

UNT

UNT(<espressione di indirizzo>)

```
PRINT UNT(&FF66)
-154
```

COMANDO: Restituisce un intero compreso tra -32768 e +32767 che rappresenta il complemento a due equivalente al valore senza segno dell'<espressione di indirizzo>.

Parole chiave associate: CINT, FIX, INT, ROUND.

UPPER$

UPPER$(<espressione di stringa>)

```
10  CLS:a$="ma quanto sei cresciuta!"
20  PRINT UPPER$(a$)
run
```

FUNZIONE: Restituisce una nuova espressione di stringa che risulta essere una copia dell'<espressione di stringa> specificata in cui tutti i caratteri alfabetici compresi tra A e Z vengono convertiti in lettere maiuscole. La funzione risulta utile per utilizzare dati in ingresso che possono essere forniti in caratteri maiuscoli o minuscoli.

Parole chiave associate: LOWER$.

USING

(Si veda PRINT USING).

VAL

VAL(<espressione di stringa>)

```
10 CLS:PRINT "Conosco le tabelline!"
20 PRINT:PRINT "Premi un tasto (da 1 a 9)"
30 a$=INKEY$:IF a$="" THEN 30
40 n=VAL(a$):IF n<1 OR n>9 THEN 30
50 FOR x=1 TO 12
60 PRINT n;"X";x;"=";n*x
70 NEXT:GOTO 20
run
```

FUNZIONE: Restituisce il VALore numerico (compresi il segno di negazione e il punto decimale) del primo carattere (o dei primi caratteri) nella <espressione di stringa> specificata.

Se il primo carattere non è un numero, viene restituito 0. Se il primo carattere è un segno di negazione o un punto decimale seguito da caratteri non numerici, verrà restituito un messaggio di errore(13) "Type mismatch".

Parole chiave associate: STR$.

VPOS

VPOS(#<canale>)

```
10 MODE 1:BORDER 0:LOCATE 8,2
20 PRINT "usa i tasti cursore su/giu"
30 WINDOW 39,39,1,25:CURSOR 1,1
40 LOCATE 1,13
50 IF INKEY(0)<>-1 THEN PRINT CHR$(11);
60 IF INKEY(2)<>-1 THEN PRINT CHR$(10);
70 LOCATE #1,3,24
80 PRINT #1,"cursore di testo ";
90 PRINT #1,"posizione verticale =";
100 PRINT #1,VPOS(#0):GOTO 50
run
```

FUNZIONE: Restituisce la corrente posizione verticale del cursore di testo in relazione all'estremità superiore della finestra di testo. Il <canale > DEVE essere specificata e NON assume il valore #0 per difetto.

Parole chiave associate: POS, WINDOW.

WAIT

WAIT <numero porta>,<maschera>[,<inversione>]

```
WAIT &FF34,20,25
```

COMANDO: Attende finchè il <numero porta> di I/O specificato non restituisce un particolare valore compreso tra 0 e 255. BASIC resta in ciclo durante la lettura della porta di I/O. Il valore letto viene sottoposto ad un'operazione di OR esclusivo con l'<inversione> e quindi ad un'operazione di AND con la <maschera> finchè non si ottiene un risultato diverso da zero.

BASIC attenderà indefinitamente fino al verificarsi della condizione richiesta.

Non è un comando da utilizzare con leggerezza.

Parole chiave associate: INP, OUT.

WEND

WEND

```
WEND
```

COMANDO: Segnala la fine del corpo di un programma che deve essere eseguito all'interno di un ciclo WHILE. WEND seleziona automaticamente il comando WHILE cui deve essere associato.

Parole chiave associate: TIME, WHILE.

WHILE

WHILE <espressione logica>

```
10 CLS:PRINT "Dieci secondi di timer":t=TIME
20 WHILE TIME<t+3000
30 SOUND 1,0,100,15
40 WEND:SOUND 129,40,30,15
run
```

COMANDO: Esegue ripetutamente il corpo di un programma mentre una data condizione risulta vera. Il comando WHILE definisce l'inizio del ciclo e specifica la condizione nell'<espressione logica>.

Parole chiave associate: TIME, WEND.

WIDTH

WIDTH <espressione intera>

```
WIDTH 40
```

COMANDO: Indica a BASIC quanti caratteri per linea devono essere stampati quando si collega una stampante. BASIC invierà quindi i caratteri di ritorno carrello o line feed quando necessario.

Il computer assume un valore per difetto di 132, a meno che non venga specificato un comando WIDTH.

Il comando WIDTH 255 elimina i caratteri di ritorno carrello e line feed, permettendo alle linee di stampa di essere "spezzate" dalla stampante. Si noti che i caratteri di ritorno carrello e line feed verranno comunque generati da comandi PRINTnon chiusi da un punto e virgola o da una virgola.

Parole chiave associate: POS.

WINDOW

WINDOW[#<canale>,]<sinistra>,<destra>,<alto>,<basso>

```
10  MODE 0:BORDER 0:REM carta di prova
20 INK 0,0:INK 1,25:INK 2,23:INK 3,21
30 INK 4,17:INK 5,6:INK 6,2:INK 7,26
40 PAPER 0:CLS
50 PAPER 1:WINDOW 2,4,1,18:CLS
60 PAPER 2:WINDOW 5,7,1,18:CLS
70 PAPER 3:WINDOW 8,10,1,18:CLS
80 PAPER 4:WINDOW 11,13,1,18:CLS
90 PAPER 5:WINDOW 14,16,1,18:CLS
100  PAPER 6:WINDOW 17,19,1,18:CLS
110  PAPER 7:WINDOW 2,19,19,25:CLS
120  GOTO 120
run
```

COMANDO: Specifica le dimensioni di un canale del testo (WINDOW - finestra) sullo schermo. I valori dei parametri <sinistra>, <destra>, <alto> e <basso> dovrebbero corrispondere alle posizioni interne dei caratteri di schermo compatibilmente con il modo di schermo in uso.

Se il <canale> non è specificato, BASIC assume come valore per difetto il flusso #0.

Ulteriori informazioni relative alle finestre si trovano nella parte 2 del capitolo "A vostra disposizione".

Parole chiave associate: WINDOW SWAP.

WINDOW SWAP

WINDOW SWAP <canale>,<canale>

```
10 MODE 1:INK 1,24:INK 2,9:INK 3,6
20 WINDOW 21,40,13,25:PAPER 3
30 WINDOW #1,1,20,1,12:PAPER #1,2
40 CLS:PRINT " finestra numero 0"
50 CLS #1:PRINT #1," finestra numero 1"
60 LOCATE 1,6
70 PRINT " finestra rossa (0)";SPC(2)
80 LOCATE #1,1,6
90 PRINT " finestra verde (1)"
100  FOR t=1 TO 1000:NEXT
110  WINDOW SWAP 0,1:GOTO 60
run
```

COMANDO: Scambia la finestra di testo specificata nel primo <canale> con quella specificata nel secondo <canale>.

Devono essere specificate entrambi i <canali>, e in questo caso NON devono essere precedute da un direttore di flusso #.

Il comando può essere utilizzato per ridirigere i messaggi prodotti da BASIC, che in genere vengono inviati al canale #0.

Ulteriori informazioni relative alle finestre si trovano nella parte 2 del capitolo "A vostra disposizione".

Parole chiave associate: WINDOW.

WRITE

WRITE [#<canale>,][<lista scrittura>]

```
10  REM scrive variabili su disco
20  INPUT "dammi una variabile numerica";a
30  INPUT "dammi una variabile di stringa";a$
40  OPENOUT "filedati"
50  WRITE #9,a,a$
60  CLOSEOUT:PRINT "i dati sono memorizzati su disco"
run
```

COMANDO: Scrive i valori degli elementi della <lista scrittura> sul canale specificato nell'espressione <canale>. Gli elementi scritti vengono separati da virgole; le stringhe sono inserite tra doppi apici.

In questo esempio i valori delle variabili inserite vengono scritti sul canale #9 (il canale di disco).

Per richiamare i valori di queste variabili dal disco è necessario usare un programma come il seguente:

```
10  REM recupera variabili da disco
20 OPENIN “filedati”:INPUT #9,a,a$
30 CLOSEIN:PRINT “I 2 valori sono:”
40 PRINT:PRINT a,a$
run
```

Parole chiave associate: INPUT, LINE INPUT.

XOR

<argomento> XOR <argomento>

```
IF “alan”<”bob” XOR “gatto”>”cane” THEN PRINT “corretto” ELSE PRINT “sbagliato”
sbagliato

IF “bob”<”alan” XOR “cane”>”gatto” THEN PRINT “corretto” ELSE PRINT “sbagliato”
sbagliato

IF “alan”<”bob” XOR “cane”>”gatto” THEN PRINT “corretto” ELSE PRINT “sbagliato”
corretto

....

PRINT 1 XOR 1
 0
PRINT 0 XOR 0
 0
PRINT 1 XOR 0
 1
```

OPERATORE: Esegue operazioni booleane bit a bit su interi. Il risultato è 1 a meno che gli argomenti siano uguali (O esclusivo).

Ulteriori informazioni relative alla logica si trovano nella parte 2 del capitolo “A vostra disposizione...”.

Parole chiave associate: AND, OR, NOT.

XPOS

XPOS

```
10 MODE 1:DRAW 320,200
20 PRINT "POSizione X cursore grafico=";
30 PRINT XPOS
run
```

FUNZIONE: Restituisce la corrente POSizione orizzontale (X) del cursore grafico.

Parole chiave associate: MOVE, MOVER, ORIGIN, YPOS.

YPOS

```
10 MODE 1:DRAW 320,200
20 PRINT "POSizione Y cursore grafico=";
30 PRINT YPOS
run
```

FUNZIONE: Restituisce la corrente POSizione verticale (Y) del cursore grafico.

Parole chiave associate: MOVE, MOVER, ORIGIN, XPOS.

ZONE

ZONE <espressione intera>

```
10 CLS:FOR z=2 TO 20
20 ZONE z
30 PRINT "X","ZONA X =";z:NEXT
run
```

COMANDO: Modifica le dimensioni della zona di stampa (specificata nelle istruzioni PRINT mediante una virgola tra gli elementi). Il valore per difetto della zona di stampa è di 13 colonne, ma può essere modificata come indicato nella <espressione intera>, che deve essere compresa tra 1 e 255.

Parole chiave associate: PRINT.

Capitolo 4
Uso dei dischi (solo 6128)

Parte 1: I dischi

Come usare i dischi

Questo paragrafo indica come usare i dischi ed introduce alcune caratteristiche di CP/M e i programmi di utilità.

Argomenti trattati:

* Come effettuare copie di riserva dei dischi originali
* Come iniziare l'esecuzione di CP/M Plus
* Come usare i file di aiuto (Help)
* Operazioni su drive singoli e multipli
* Copia di file con PIP
* Operazioni con i dischi di dati di BASIC
* Applicazioni BASIC ad avvio automatico
* Come installare una applicazione CP/M Plus ad avvio automatico

La parte 7 del corso Fondamenti ha descritto come formattare un disco di sistema vergine che può essere usato con BASIC e i giochi oltre che con CP/M.

La parte 10 del corso Fondamenti ha descritto come fare copie dei dischi usando il programma DISCKIT3 (sul lato 1 dei dischi di sistema).

Questo paragrafo considera come usare i dischi a seconda dei programmi che si vogliono memorizzare.

Copie di riserva dei dischi originali

E' molto importante fare copie di riserva del disco di sistema fornito con il computer e poi tenere l'originale in un luogo sicuro: è infatti molto costoso sostituirli se danneggiati! Si ricorda che ogni disco fornito ha due facce.

Il Side 1 (lato 1) è molto importante: contiene la copia originale del CP/M Plus ed un insieme di programmi di utilità per la gestione dei dischi. Il Side 2 (lato 2) contiene file per i programmatori in assembler.

Occorre pensare alle copie dei dischi originali come ad una 'libreria' di programmi. Di solito, si selezionerà il programma desiderato, invece di copiare il programma su un disco vuoto per poi eseguirlo da lì, inserendo il 'disco di libreria' sul quale questo è memorizzato.

Ancora una volta occorre ricordare che i 'dischi di libreria' che si usano DEVONO ESSERE COPIE, fatte dai dischi originali forniti con il computer.

Si ricorda inoltre che se si sta usando in fase di copia un disco vergine come destinazione, il programma DISCKIT (sul lato 1) lo formatterà in tale fase.

Iniziamo ad usare CP/M Plus

Si è soliti vedere, quando si accende il 6128, il BASIC AMSTRAD.

Questo resterà in memoria fino a quando non viene sostituito da un programma BINario eseguito da AMSDOS o caricando CP/M Plus mediante il comando **|CPM**.

Una volta caricato il sistema operativo CP/M Plus il 6128 non dovrà più fare accesso al lato 1 del disco; a meno che, naturalmente, non si voglia eseguire un programma di utilità contenuto in esso. Solo i dischi ad avvio automatico devono essere dischi di sistema (System); tutti gli altri dischi possono essere dischi di dati (Data) i quali hanno una capacità di memorizzazione leggermente maggiore.

Eseguire un programma consiste semplicemente nell'inserire il disco contenente il programma e scriverne il nome. I dati usati da un programma possono essere salvati sia sul disco contenente il programma sia su un altro. CP/M Plus permette, così come AMSDOS, di cambiare il disco. Se alcuni programmi, contenenti alcune utilità, devono essere per comodità presenti sul disco, si usi il programma PIP sul lato 1 per copiarli come descritto in questo capitolo e nel Capitolo 5.

Come tenere il file PROFILE!

Il disco di sistema fornisce un file speciale, il cui nome è PROFILE.SUB, che contiene una lista di comandi che devono essere eseguiti automaticamente quando viene caricato CP/M Plus. E' possibile, quindi, (se non lo si è già fatto) inserire una COPIA del lato 1 (Side 1) dei dischi di sistema e scrivere al prompt A>:

REN PROFILE.SUB=PROFILE.ENG

...che creerà il file PROFILE.SUB dal PROFILE.ENG. Questo file, che verrà eseguito al prossimo caricamento di CP/M Plus, contiene i comandi:

SETKEYS KEYS.CCP
LANGUAGE 3

...per far sì che i tasti cursore siano disponibili nella scrittura dei comandi CP/M, e per convertire l'output su schermo in caratteri inglesi (da quelli americani), in modo che premendo **[SHIFT]3** si visualizzi il segno '£'.

Quando la tastiera è stata fissata com il comando SETKEYS KEYS.CCP le linee di comando CP/M possono essere editate nello stesso modo di quelle BASIC. Per ulteriori dettagli si consulti la parte 2 del Capitolo 5.

Un drive o due?

Quando viene inizialmente caricato CP/M Plus, questo scopre il numero di drive connessi. Questo numero viene visualizzato in uno dei messaggi iniziali. Si noti che questo processo può essere eluso se il secondo drive ha un disco inserito parzialmente.

Tutti i messaggi di errore relativi ai dischi vengono visualizzati nella venticinquesima linea dello schermo. I programmi stessi usano solo ventiquattro linee.

Quando si ha solo un drive, la linea finale visualizza anche il messaggio 'Drive is A:' o 'Drive is B:'. Questo perchè CP/M Plus permette di lavorare con un meccanismo fisico come se se ne avessero due. Si avranno due dischi da alternare e la linea finale dello schermo indicherà quando inserire i dischi a seconda delle richieste del programma. Questo metodo permette di non acquistare il secondo drive ma richiede spesso di cambiare i dischi: ciò è una perdita di tempo e induce alla possibilità di errori umani.

Copia di file da un disco in un altro

Il programma di utilità PIP (Peripheral Interchange Program) viene fornito per copiare i file da un disco in un altro.

Prima di tutto occorre caricare PIP dal lato 1 (Side 1), scrivendo, al prompt A>:

PIP

...il prompt * indicherà che PIP è stato correttamente caricato. Di solito si copieranno i file dal disco origine (Source) (nel Drive A:) in quello di destinazione (Destination) (nel Drive B:). Abbiamo già visto questo processo su un sistema a singolo drive; il Drive A: e il Drive B: coincidevano (erano cioè lo stesso meccanismo).

Per copiare un file, ad esempio SUBMIT.COM, dopo il prompt * si scriva:

B:=A:SUBMIT.COM

Per copiare tutti i file dal disco origine in quello destinazione si usa il comando:

B:=*.*

Per uscire da PIP, al prompt *, si preme **[RETURN].**

PIP è un programma molto sofisticato; ulteriori dettagli sul suo uso sono descritti nel Capitolo 5.

Dischi di dati di BASIC

Come appena descritto un disco di sistema (System) viene usato di solito solo come disco di avvio CP/M. I dischi usati per BASIC possono essere del resto solo di dati (Data-Only) i quali hanno una capacità di memorizzazione maggiore.

Tale disco deve essere formattato usando il programma DISKIT3. Per copiare i programmi su questo tipo di disco occorre usare PIP (caricato dal lato 1 (Side 1)) oppure usare i comandi LOAD e SAVE da BASIC.

Dischi BASIC ad avvio automatico

Se si acquista un programma applicativo scritto in BASIC AMSTRAD per il 6128 questo dovrebbe essere eseguito automaticamente all'accensione del computer. Come con i dischi di sistema originali forniti con il 6128 anche con questi occorre mantenere gli originali al sicuro ed usare le copie.

Dischi CP/M ad avvio automatico

Il sistema operativo CP/M permette di caricare ed eseguire una enorme libreria software scritta per i personal computer che usano CP/M. La 'logica' fondamentale di tali programmi è stata già disposta: tutto ciò che è richiesto per usarli sul 6128 è di porli in un disco disponibile e forse di informarli di un metodo particolare che il 6128 usa per far riferimento allo schermo.

Un insieme di programmi di un disco progettato per eseguire una specifica applicazione è detto 'pacchetto'. Questi pacchetti sono, di solito, progettati per operare su una vasta gamma di computer, ognuno dei quali ha una propria dimensione di schermo ed un proprio modo di spostare il cursore.

Il 6128, quando esegue programmi di CP/M Plus, ha un 'emulatore di terminale' e le caratteristiche sono diverse dai codici di controllo del BASIC.

A volte, i pacchetti acquistati sono già stati 'installati' per i sistemi AMSTRAD, oppure è stata predisposta una installazione compatibile con il 6128. Si seguano semplicemente, se disponibili, le istruzioni fornite con il software per un protocollo Zenith Z19/Z29. Se il pacchetto non ha incluso tale protocollo nè ha delle specifiche riguardanti AMSTRAD, si consulti il paragrafo 'Configurazione di un programma CP/M' che indica alcuni comandi che possono essere inviati al 6128 per produrre gli effetti che il pacchetto richiede. Di solito la procedura di installazione richiede la scrittura di codici quando richiesti. Si seguano quindi le istruzioni fornite con il pacchetto.

Il software acquistato deve essere fornito su un disco usabile dal sistema. I computer usano diversi tipi di dischi: possono avere la stessa dimensione ma ciò non significa necessariamente che siano compatibili fra loro. Si richieda al proprio fornitore la versione AMSTRAD .

Come creare un disco CP/M ad avvio automatico

Come per i programmi applicativi stessi è spesso utile avere le utilità SETKEYS.COM e SUBMIT.COM (con i loro file di istruzione associati) su un disco ad avvio automatico.

Per trasferire i file **.COM** e i file di istruzione di **SUBMIT** è possibile usare PIP. In quest'ultimo modo **PIP** è effettivamente un editor linea a linea. Il file **PROFILE.ENG**, ad esempio, sul lato 1 (Side 1) potrebbe esere stato creato usando i seguenti comandi:

(Inserire il disco di sistema, Side 1 nel Drive A:). Si scriva:

PIP

(Togliere il disco di sistema ed inserire il disco destinazione). Si scriva:

PROFILE.ENG=CON:
SETKEYS KEYS.CCP
[CONTROL]J LANGUAGE 3
[CONTROL]Z

Configurazione di un programma CP/M

Il 6128 ha una ampia gamma di codici di controllo disponibili per installare il pacchetto software in modo sia eseguibile con CP/M. Molti processi sui dati e molti pacchetti richiedono la visualizzazione di messaggi e l'input in qualsiasi punto dello schermo, la comprensione di codici di controllo.

Il pacchetto che si possiede è già stato configurato per il sistema AMSTRAD, non occorre quindi fare nessuna operazione di installazione.

Configurazione dell'output da un pacchetto

La procedura di installazione di un pacchetto consiste di solito nell'esecuzione di un programma speciale (spesso chiamato **INSTAL**) che fa diverse domande inerenti i parametri dello schermo del 6128. Le risposte devono essere rilevate dalla tabella che segue che è una sintesi del Capitolo 6 parte 15.

Codici di controllo	Esadecimale	Decimale	Operazione
[BEL]	&07	7	Emette un suono
[BS]	&08	8	Sposta il cursore indietro di una posizione
[LF]	&0A	10	Sposta il cursore in giù di una linea
[CR]	&0D	13	Sposta il cursore nell'angolo a sinistra di una finestra sulla linea attuale
[ESC]A	&1B &41	27 65	Sposta il cursore in sù di una linea
[ESC]C	&1B &43	27 67	Sposta il cursore in avanti di una posizione
[ESC]E	&1B &45	27 69	Pulisce lo schermo
[ESC]H	&1B &48	27 72	Cursore nell'angolo in alto a sinistra dello schermo
[ESC]J	&1B &4A	27 74	Pulisce lo schermo dalla posizione attuale del cursore fino alla fine dello schermo
[ESC]K	&1B &4B	27 75	Pulisce lo schermo dalla posizione attuale del cursore fino all'angolo in alto a destra
[ESC]L	&1B &4C	27 76	Inserisce una linea
[ESC]M	&1B &4D	27 77	Cancella una linea
[ESC]N	&1B &4E	27 78	Cancella un carattere nella posizione attuale dello schermo
[ESC]Y	&1B &59	27 89	
	<c> <r>	<c> <r>	Sposta il cursore sullo schermo nella posizione indicata. <c> è la colonna + 32, <r> è la riga + 32
[ESC]d	&1B &64	27 100	Pulisce lo schermo dalla posizione del cursore inclusa fino alla fine
[ESC]o	&1B &6F	27 111	Pulisce lo schermo dall'angolo in alto a sinistra fino alla posizione del cursore inclusa
[ESC]p	&1B &70	27 112	Entra in modalità video inverso
[ESC]q	&1B &71	27 113	Esce dalla modalità video inverso

Configurazione dell'input da un pacchetto

I programmi del pacchetto si attenderanno di poter richiedere informazioni alla tastiera. Molti dei tasti della tastiera del 6128 restituiscono valori standard ad eccezione dei tasti cursore. E' possibile usare l'utilità SETKEYS per ridefinire questi codici prodotti dalla tastiera sebbene, dove è possibile, è preferibile per ogni pacchetto essere configurato per accettare valori standard.

Sfortunatamente non vi è una regola generale tra i vari software nell'uso delle parole chiave. I caratteri 'spazio', **[TAB]** e **[RETURN]** sono praticamente universali ma il tasto 'backspace' è spesso diverso! Si provi, ad esempio, a confrontare i diversi codici di controllo relativi all'operazione 'sposta il cursore all'inizio della linea':

in CP/M :**[CONTROL]B**

e nei word processor standard: **[CONTROL]Q S**

Sono forniti come standard tre utili insiemi di codici. Ogni configurazione può essere richiamata dal file contenuto nel lato 1 (Side 1) dei dischi di sistema:

SETKEYS KEYS.CCP

... già descritto come uno dei comandi eseguito automaticamente da PROFILE.SUB che fissa la tastiera in modo possa usare i comandi CP/M.

Esecuzione di un pacchetto ad avvio automatico di CP/M

Di solito tutto ciò che è richiesto è di scrivere il nome del programma principale del pacchetto al prompt A>. Per eseguire ad esempio il programma PAYROLL.COM, si scrive semplicemente:

PAYROLL

Se è richiesta una qualsiasi configurazione verrà forse fornito un file.

Avvio automatico di una pacchetto CP/M

E' possibile modificare il sistema operativo CP/M in modo che esegua automaticamente un programma particolare all'inizio del disco di sistema. E' possibile ottenere ciò includendo il nome del programma alla fine del file PROFILE.SUB di quel disco.

Capitolo 5
AMSDOS e CP/M (solo 6128)

Parte 1: AMSDOS

Argomenti trattati:

* Introduzione a AMSDOS
* Directory su disco
* Cambio di disco
* Nomi di file e tipi di file
* Intestazioni AMSDOS
* Nomi di file su sistema a due drive
* Caratteri jolly
* Programma di esempio sull'uso dei comandi AMSDOS
* Elenco dei comandi AMSDOS
* Trattamento e copia di file
* Guida di riferimento dei messaggi di errore

Introduzione

AMSDOS estende il BASIC AMSTRAD fornito con il proprio computer fornendo diversi comandi esterni che sono identificati dal simbolo | che li precede.

AMSDOS permette all'utente di cambiare i dischi liberamente sempre che non vi siano file in uso; nel qual caso verrà visualizzato un messaggio di errore e vi potrebbe essere una perdita di dati se il file aperto era in fase di scrittura.

Directory su disco

Ogni disco è suddiviso in due parti, la directory e l'area dati. La directory contiene una lista di tutti i nomi di file ed una "mappa" per il loro reperimento. AMSDOS o CP/M possono calcolare la dimensione di un particolare file guardando la directory. Il calcolo dello spazio disponibile su disco viene fatto sommando la dimensione di tutti i file e guardando quanto spazio resta inutilizzato.

Quando viene letto un file, viene esaminata la voce nella directory e fornita la locazione su disco. Quando viene creato un nuovo file, viene allocato uno spazio libero e quando viene cancellato tale spazio viene abbandonato. La directory lavora con unità di 1K e può trattare un massimo di 64 voci. I file larghi hanno una voce ogni 16K, sebbene questo aspetto sia nascosto all'utente.

Cambio di dischi

In AMSDOS (e CP/M Plus) è possibile cambiare disco o rimuoverlo sempre che il drive non vi stia accedendo e che nessun file sia aperto per lettura o scrittura.

Il cambio di disco mentre è in fase di scrittura può danneggiare i dati presenti sul disco. Se un disco viene cambiato mentre vi è ancora un file aperto, quando AMSDOS lo scopre lo abbandona e produce un messaggio di errore. Qualsiasi dato ancora da scrivere sul disco verrà perso e l'ultima voce della directory non verrà scritta. Comunque, AMSDOS può scoprire tale cambiamento quando legge la directory e cioè ogni 16K (e quando il file viene aperto o chiuso). Così, potenzialmente 16K di dati potrebbero essere danneggiati cambiando il disco su cui vi è ancora un file aperto.

Nomi di file e tipi di file di AMSDOS

E' pratica comune fornire un nome al file in modo che sia chiaro di che tipo si tratta. Il nome NON costringe il computer ad usare il file in un modo particolare, comunque alcuni programmi accetteranno un file quando ha un corretto tipo di nome. AMSDOS accetterà qualsiasi nome, ma cercherà preferibilmente alcuni tipi di file a meno che non venga specificato altrimenti (si veda "Intestazioni AMSDOS" più avanti).

Nomi di file

Il nome del file è costituito da due parti suddivise da un . (punto). La prima parte può essere costituita da un massimo di otto caratteri la seconda da tre. Quindi "ROINTIME.DEM", "DISCKIT3.COM" e "DISC.BAS" sono nomi di file legali.

La seconda parte del nome del file è detta tipo di file. I nomi di file e i tipi possono essere composti da lettere e numeri, ma non sono permessi spazi e simboli di punteggiatura. Alcuni tipi di file sono:

.<spazio> Tipo non specificato. Può essere un file di dati creato da un comando OPENOUT "nomefile" o un programma BASIC salvato da AMSDOS usando il comando SAVE "nomefile",A.

.BAS Un programma BASIC salvato da AMSDOS usando il comando SAVE "<nomefile>" o SAVE "<nomefile>,P o SAVE "<nomefile>.BAS",A.

.BIN Un programma o una area di memoria salvata da AMSDOS usando SAVE "<nomefile>",B,<parametro binario>.

.BAK Una vecchia versione di un file dove AMSDOS o un programma di utilità ha salvato una nuova versione di un file usando un nome esistente. Ciò permette all'utente di rintracciare, se necessario, la versione precedente.

.COM Un file di comandi. I comandi di utilità di CP/M sono tutti di questo tipo.

.SUB Un file di istruzioni per il programma CP/M SUBMIT

Intestazioni di AMSDOS

AMSDOS salva automaticamente i file con un identificatore di tipo, quindi non è necessario specificarne uno a meno che non si voglia tener conto di quelli descritti precedentemente. I file di programma BASIC, protetti e non, e i file binari sono salvati su disco con una intestazione (header) e quindi il comando AMSDOS:

LOAD "<nomefile>"

... può riconoscerli ed agire di conseguenza. Se il comando LOAD di AMSDOS non trova

l'intestazione assume che sia un programma ASCII, ad esempio un file di testo. Nonostante i contenuti dell'intestazione quando viene richiesto ad AMSDOS di caricare un file senza specificare il tipo di file, esso controlla prima un file di tipo:

.<spazio>

Se non esiste cerca un file di tipo:

.BAS

... ed infine uno di tipo:

.BIN

Ciò permette all'utente di abbreviare il nome del file, non è cioè necessario specificare il tipo di file ogni volta.

Un file di dati aperto con il comando OPENOUT e successivamente scritto non avrà intestazione e i contenuti dei comandi WRITE, PRINT o LIST di BASIC saranno in ASCII, ovvero testo. Il comando OPENIN, nel caso non venga specificato tipo di file, cercherà tale file con lo stesso ordine di LOAD.

Nomi di file su un sistema a due drive

Su un sistema a due drive, un sistema cioè a cui è stato connesso un drive addizionale, i file possono essere esistenti su entrambi i drive. Il computer non guarderà automaticamente su entrambi i drive; l'utente quindi deve specificare quale vuole usare. E' possibile usare il comando |A o |B o |DRIVE (la descrizione completa viene fornita più avanti) per selezionare un drive e successivamente usare un normale nome di file o alternativamente scegliere un altro drive per difetto specificando A: o B: come prefisso del nome del file. Così per esempio:

```
|B
SAVE "PROG.BAS"
|A
```

... e ...

```
|A
SAVE "B:PROG.BAS"
```

... salvano il programma sul secondo drive, il Drive B.

In modo analogo, è possibile scegliere un altro numero USER (compreso tra 0 e 15: i numeri USER permettono di suddividere la directory) specificandolo come prefisso del nome del file. Così, per esempio:

LOAD "15:PROG.BAS"

... e ...

SAVE "15:PROG.BAS"

... dovrebbero caricare e salvare il programma nella sezione di disco con numero USER 15, senza considerare il parametro per difetto USER (si veda il comando |USER).

E' infine possibile scegliere entrambi i parametri USER e DRIVE (in quell'ordine) specificandoli insieme al prefisso del nome del file, ad esempio:

RUN "15B:PROG.BAS"

Caratteri jolly

Viene spesso richiesto di eseguire alcune operazioni (copia, cancellazione, ecc.) su più file. Quando viene richiesto, in una particolare operazione, un nome di file, AMSDOS scandisce tutta la directory alla ricerca di quel nome. E' possibile, comunque (quando il comando lo permette) eseguire l'operazione su un gruppo di file dove alcuni caratteri all'interno del nome del file "non hanno importanza". Ciò è mostrato dall'uso del carattere ? nella posizione "senza importanza". Se il blocco vuoto (o quello che resta dell'intero blocco) di una qualsiasi parte del nome del file non ha importanza, allora il blocco di ? può essere abbreviato dal simbolo *. Così, per esempio, FRED* è una abbreviazione di FRED.??? e F*.BAS è una abbreviazione di F??????.BAS

Infine l'espressione *.* significa "tutti i file".

esempi:

DIRECTORY	CON *. BAS	CON FRED?. BAS	CON F*. BA?
BERT.BAS	BERT.BAS		
FRED1.BAS	FRED1.BAS	FRED1.BAS	FRED1.BAS
FRED2.BAS	FRED2.BAS	FRED2.BAS	FRED2.BAS
FRED3.BAK			FRED3.BAK
FRED3.BAS	FRED3.BAS	FRED3.BAS	FRED3.BAS
FINISH.BAS	FINISH.BAS		FINISH.BAS

Esempi dell'uso dei comandi AMSDOS in un programma

Per fornire una buona conoscenza dei comandi AMSDOS raccomandiamo di seguire gli esempi e leggere i prossimi paragrafi di questo capitolo. NON si scriva o esegua questi programmi con uno dei dischi di sistema CP/M originale inseriti.

Salvataggio di variabili e copia di uno schermo

Il seguente esempio di programma scrive su disco e vi sarà quindi bisogno di un dischetto vuoto (formattato) o di un disco di lavoro inserito per eseguire il programma. Il programma disegna la bandiera inglese e salva l'intero schermo su disco.

```
10 dumpfile$="flagdump.srn"
20 MODE 1:BORDER 0
30 DIM colour(2)
40 FOR i=0 TO 2
50 READ colour(i): REM ottiene i colori da DATA
60 INK i,colour(i)
70 NEXT
80 ON ERROR GOTO 430
90 OPENIN "param.dat"  ' testa se il file esiste
100 CLOSEIN:ON ERROR GOTO 0
110 IF errnum=32 AND DERR=146 THEN CLS: GOTO 160 ' il file non esiste
120 CURSOR 1:PRINT "Vuoi scrivere sul vecchio file? S/N);
130 a$=INKEY4:ON INSTR("  SN",UPPER$(a$)) GOTO 130,150,140: GOTO
130
140 PRINT a$:PRINT "Programma abbandonato":END
150 PRINT a$:CURSOR 0
160 OPENOUT "param.dat"
170 WRITE #9,dumpfile$,1:REM  salva il nomefile e il mode
180 FOR i=0 TO 2
190 WRITE #9,colour(i): REM salva i colori
200 NEXT i
210 CLOSEOUT
220 CLS
230 gp=1:GRAPHICS PEN gp:w=125
240 x=-65:a=240:y=400:b=-150:GOSUB 400
```

continua nella prossima pagina

```
250 y=0:b=150:GOSUB 400
260 x=575:a=-240:y=400:b=-150:GOSUB 400
270 y=0:b=150:GOSUB 400
280 gp=2:GRAPHICS PEN gp:w=40*290 a=240:x=-40:y=400:b=-150:GOSUB
400
300 x=0:y=0:b=150:GOSUB 400
310 a=-240:x=640:y=0:b=150:GOSUB 400
320 x=600:y=400:b=-150:GOSUB 400
330 ORIGIN 0,0,256,380,0,400:CLG 1
340 ORIGIN 0,0,0,640,150,250:CLG 1
350 ORIGIN 0,0,280,352,0,400:CLG 2
360 ORIGIN 0,0,0,640,168,230:CLG 2
370 SAVEdumpfile $,b,&C000,&4000
380 DATA 2,26,6
390 END
400 MOVE x,y:DRAWR a,b:DRAWR w,0:DRAWR -a,-b
410 MOVE x+a/2+w/2,y+b/2:FILL gp
420 RETURN
430 errnum=ERR:RESUME NEXT
run
```

Si noti l'uso dei tipi di file .DAT e .SRN. Questi tipi di file sono usati per ricordare il contenuto del file. Il file PARAM.DAT sarà un file di dati ASCII senza intestazione, mentre FLAGDUMP.SRN un file binario con intestazione.

Si noti come il programma cerchi deliberatamente di leggere dal file PARAM.DAT prima di scrivervi al fine di stabilire se il file esiste già. Se non esiste viene riportato un errore; l'errore viene trattato dal programma e l'esecuzione procede senza interruzioni. Se il file esiste non viene riportato nessun errore ed il programma chiede automaticamente se si vuole riscrivere sul file.

I particolari della copia dello schermo, cioè la modalità di schermo, la tavolozza dei colori e il nome del file contenente l'informazione attuale, vengono salvati in un file di parametri. Ciò illustra l'uso dei file di dati per scrivere le variabili di un programma (dumpfile$) e le costanti (1) salvandole per usarle con un altro programma.

Caricamento di uno schermo

Il seguente programma di esempio è un programma multi uso che usa un file di parametri per controllare la sua azione. Si noti come le variabili sono inserite dal file di dati con la funzione EOF che permette una variazione automatica della dimensione del file. E' importante che lo schermo visualizzato dal programma sia stato salvato in una posizione nota in memoria, altrimenti il risultato sarà deviato. Ciò è assicurato salvando il programma con un comando MODE e stando attenti a non far scrollare lo schermo.

```
10 DIM colour(15): REM Preparativi per 16 colori
20 OPENIN "param.dat"
30 INPUT #9,nomefile$,screenmode
40 i=0
50 WHILE NOT EOF
60 INPUT #9,colour(i)
70 INK i,colour(i)
80 i=i+1
90 WEND
100 CLOSEIN
110 MODE screenmode:BORDER 0
120 LOAD nomefile$
run
```

Elenco dei comandi esterni AMSDOS

|A

|A

COMANDO: Fissa come drive per difetto il Drive A. E' equivalente a |DRIVE con parametro A (Il drive principale di un computer è il Drive A).

|B

|B

COMANDO: Fissa come drive per difetto il Drive B. E' equivalente a |DRIVE con parametro B (Il drive principale di un computer è il Drive A).

|CPM

|CPM

COMANDO: Commuta su un ambiente di disco alternativo caricando il sistema operativo dal disco di sistema. Il sistema operativo fornito con il computer è CP/M Plus e CP/M 2.2

Questo comando non avrà esito positivo se il drive non contiene un disco di sistema con la copia di CP/M sul lato 1 (Side 1) che caricherà CP/M Plus mentre le copie del lato 4 (Side 4) caricheranno il vecchio sistema operativo CP/M 2.2.

|DIR

|DIR[,<espressione stringa>]

|DIR,"*.BAS"

COMANDO: Visualizza la directory del disco (in classico modo CP/M) e lo spazio libero. Se l'espressione stringa viene omessa viene assunto il carattere jolly *.*.

|DRIVE

|DRIVE, <espressione stringa>

|DRIVE,"A"

COMANDO: Fissa il drive per difetto. Questo comando non avrà successo se AMSDOS non è in grado di leggere il disco nel drive richiesto.

|ERA

|ERA, <espressione stringa>

|ERA,"*.BAK"

COMANDO: Cancella tutti i file uguali al nome campione e che non siano a sola lettura (Read/Only). I caratteri jolly sono permessi.

|REN

|REN,<espressione stringa>,<espressione stringa>

|REN,"NUONOME.BAS","VECNOME.BAS"

COMANDO: Fornisce un nuovo nome ad un file. Il nuovo nome scelto deve essere diverso dai nomi di file già esistenti. I caratteri jolly sono permessi.

Il parametro USER può essere specificato all'interno della espressione stringa. Il comando, ad esempio |REN,"0:NUO.BAS","15:VEC.BAS" fornirà un nuovo nome al file in USER 15 chiamato "VEC.BAS" chiamandolo "NUO.BAS" in USER 0 senza tener conto dei parametri per difetto o di parametri definiti precedentemente da USER.

|USER

|USER,<espressione intera>

|USER,3

COMANDO: Determina su quale delle 16 sezioni di una directory (tra 0 e 15) devono essere eseguite le funzioni di disco (cioè CAT, LOAD, |DIR ecc).

E' possibile trasferire, mediante comando |REN, un file di un numero USER su un altro. Ad esempio, |REN,"15:EXAMPLE.BAS",0:EXAMPLE.BAS" trasferisce un file da un numero USER 0 ad un numero USER 15, sebbene il nome del file stesso (EXAMPLE.BAS) non sia cambiato.

Come copiare file da un disco in un altro

File AMSDOS con intestazioni

E' possibile copiare questo tipo di file in ambiente CP/M usando PIP (si veda la seconda parte di questo capitolo). Qualsiasi file creato da AMSDOS con un record di intestazione (si veda "Intestazioni di AMSDOS" descritto precedentemente) sarà interamente copiabile da disco a disco, ma in generale il contenuto del file non verrà compreso dai programmi CP/M.

File ASCII

I file creati da AMSDOS senza intestazione sono generalmente in ASCII e sono sia copiabili sia compresi dai programmi CP/M. In particolare dovrebbe essere possibile scambiare facilmente tra AMSDOS e i programmi CP/M i file di programma ASCII, i file di dati ASCII e i file di testo ASCII.

File a sola lettura (Read/Only)

E' possibile, usando CP/M, fissare qualunque file in modo sia a sola lettura e/o fissarlo su uno stato di sistema speciale nella directory. Tali attributi possono essere fissati o riavviati in ambiente CP/M, ma sono accettati da AMSDOS. Per ulteriori dettagli si veda la seconda parte di questo capitolo (Utilità SET).

Guida di riferimento dei messaggi di errore

Quando AMSDOS non può, per qualche ragione, eseguire un comando visualizza un messaggio di errore. Se il problema è di tipo hardware al messaggio di errore segue la domanda:

Retry, Ignore or Cancel?

(Ovvero: Riprova, Ignora o Annulla?)

R fa in modo che l'operazione venga ripetuta, possibilmente dopo che l'utente ha rimediato alla causa.

I fa sì che il computer continui come se il problema non sia stato incontrato, che porta spesso ad un risultato inaspettato e poco conveniente.

C fa sì che l'operazione venga annullata, il che comporta spesso un ulteriore messaggio di errore.

Significato dei messaggi di errore

Unknown command

...Il comando non è stato scritto correttamente.

Bad command

...Il comando non può essere eseguito per qualche ragione. Un errore di sintassi o una configurazione hardware inappropriata.

<nomefile> already exists

...L'utente sta cercando di fornire un nome già in uso.

<nomefile> not found

...Il file non esiste.

Drive <drive>: directory full

...Non vi è più spazio disponibile nella directory per la nuova voce.

Drive <drive>: disc full

...Non vi è più spazio su disco per i nuovi dati.

Drive <drive>: disc changed, closing <nomefile>

...I dischi sono stati cambiati con ancora file aperti.

<nomefile> is read only

...Non è possibile usare il file poichè è a sola lettura (read/Only). I file possono essere fissati, in ambiente CP/M solo a sola lettura (Read/Only) o lettura/scrittura (Read/Write).

Drive <drive>: disc missing

...Non vi è disco nel drive, o il disco non è situato ed inserito correttamente. Si consiglia di estrarlo e reinserirlo e di scrivere R.

Drive <drive>: disc is write protected

...Si è cercato di scrivere su un disco con la tacca di protezione dalla scrittura aperta. Per usare il disco occorre estrarlo e chiudere la tacca, quindi reinserire il disco e scrivere R.

Drive <drive>: read fail

...Errore di lettura su disco di tipo hardware. Si consiglia di estrarre il disco, reinserirlo e scrivere R.

Drive <drive>: write fail

...Errore di scrittura su disco di tipo hardware. Si consiglia di estrarre il disco, reinserirlo e scrivere R.

Failed to load CP/M

...Errore di lettura in fase di caricamento di CP/M in un comando |CPM, oppure non si sta usando un disco di sistema contenente CP/M valido. Si noti che cercando di caricare CP/M da un disco di dati formattato si produrrà un errore di tipo "read fail".

Parte 2: CP/M

CP/M Plus

Argomenti trattati:

* Introduzione a CP/M
* Avviamento di CP/M Plus
* Modo diretto
* Programmi transienti
* Gestione delle periferiche

CP/M Plus è un sistema operativo per la gestione dei dischi. E' un programma speciale che permette di sfruttare le reali potenzialità del 6128. I 128K di RAM vengono usati interamente con oltre 61K a disposizione per i programmi dell'utente. Il CP/M permette di accedere ai file di dati in modo diretto e l'implementazione del 6128 include un sofisticato emulatore di terminale.

Poichè CP/M è disponibile su diversi computer vi sono centinaia di programmi applicativi disponibili e molta documentazione.

Introduzione

Il sistema operativo CP/M fornisce un modo per comunicare con il computer e trattare file e periferiche. Sono disponibili comandi speciali (e programmi di utilità su disco) per aiutare ad eseguire i propri programmi applicativi.

Molti utenti desiderano conoscere quanto basta per iniziare e il resto di questo capitolo elenca tutte le caratteristiche senza punti oscuri o troppi fronzoli.

BASIC ha un suo modo diretto e il prompt "Ready", anche CP/M ha un modo diretto ed è identificato dai prompt A> o B>. Sono disponibili molti comandi interni, ma la maggior parte del lavoro consiste nel caricare ed eseguire "programmi transienti". Sono chiamati "transienti" perchè sono presenti in memoria (caricati da disco) solo al momento del loro uso.

Così come gli errori standard di CP/M, il sistema genera diversi messaggi di errore di tipo hardware che si possono distinguere dagli altri in quanto questi appaiono normalmente in fondo allo schermo sotto forma di menù a barre.

CP/M Plus su disco

La maggior parte di CP/M Plus risiede in un file speciale il cui tipo di file è ".EMS", ed è situato sul lato 1 (Side 1) del disco di sistema. Il computer carica CP/M da tale file in memoria in due processi.

Inizialmente, il comando |CPM di AMSDOS carica il primo settore di traccia 0. Su un disco di sistema questo settore è stato modificato in modo carichi il file .EMS in memoria. Le restanti tracce del sistema sono inutilizzate.

File di avvio iniziale

Durante il processo di caricamento, quando viene attivato CP/M Plus, se il file PROFILE.SUB è presente su disco, le istruzioni in quel file vengono eseguite mediante comando SUBMIT. Questa caratteristica può essere utilizzata per ridefinire la tastiera, modificare lo schermo di output, inizializzare la stampante o far eseguire automaticamente un programma applicativo. Nel capitolo 4 indichiamo come fornire un nuovo nome al file Profile presente sul lato 1 per poterlo attivare.

Mentre il file profile è in esecuzione viene aperto, sul disco, un file temporaneo; il disco deve dunque essere abilitato alla scrittura. Questo perchè il disco originale stesso

potrebbe non includere un file profile riconoscibile.
I file profile possono essere scritti usando un word processor, un editor (come ED.COM) o direttamente da BASIC. Il piccolo programma BASIC che segue potrebbe essere stato usato per generare un file PROFILE.SUB.

```
10 OPENOUT "PROFILE.SUB"
20 PRINT #9, "SETKEYS KEYS.CCP"
30 PRINT #9, LANGUAGE 3"
40 CLOSEOUT
```

Codici di controllo da console

In ambiente CP/M vengono usate molte operazioni di tasti speciali. Queste parole chiave sostituiscono l'azione del tasto **[ESC]** e dei tasti cursore usati nel BASIC AMSTRAD. I codici di controllo che seguono sono assegnati dopo l'esecuzione del comando:

SETKEYS KEYS.CCP

...dove entrambi i programmi transienti SETKEYS.COM e il file KEYS.CCP sono situati nel lato 1 (side 1) del disco di sistema.

Codice di controllo	Tasto	Azione
[CONTROL]A	**<-**	Sposta il cursore di un carattere a sinistra
[CONTROL]B	**[CONTROL]<-** o **[CONTROL]->**	Sposta il cursore all'inizio della linea. Se il cursore è già all'inizio della linea lo sposta alla fine.
[CONTROL]C	**[CONTROL][ESC]**	Abbandona.
[CONTROL]E	**[CONTROL][RETURN]**	Ritorno a capo fisico
[CONTROL]F	**->**	Sposta il cursore di un carattere a destra.
[CONTROL]G	**[CLR]**	Cancella il carattere su cui è posizionato il cursore.
[CONTROL]H	**[DEL]**	Cancella il carattere precedente il cursore
[CONTROL]I	**[TAB]**	Sposta il cursore sul punto di tabulazione successivo.
[CONTROL]J		Invia una linea di comando

[CONTROL]K	**[CONTROL][CLR]**	Cancella fino alla fine della linea.
[CONTROL]M	**[RETURN] o [ENTER]**	
[CONTROL]P		Commutatore hardcopy. Attiva o disattiva gli schermi su stampante
[CONTROL]Q		Riattiva l'output su schermo.
[CONTROL]R	**[CONTROL][ENTER]**	Riscrive la linea di comando.
[CONTROL]S	**[ESC]**	Ferma l'output su schermo da CP/M. Si usa **[CONTROL]Q** per riprenderla.
[CONTROL]U		Scarica la linea
[CONTROL]W	**[COPY]**	Richiama l'ultima linea di comando scritta.
[CONTROL]X	**[CONTROL][DEL]**	Cancella dall'inizio della linea fino alla posizione del cursore.
[CONTROL]Z		Fine testo.

Nomi di file

Molti comandi accettano come parametri nomi di file e, dove specificato, il nome file può contenere caratteri jolly (si veda il paragrafo "Caratteri jolly" nella parte 1 di questo capitolo). Tutti i nomi di file verranno commutati in maiuscolo.

Nei comandi diretti e nei programmi di utilità NON sono richieste le doppie virgolette intorno al nome del file. Si ricorda che i nomi di file possono avere come prefisso sia A: che B: per forzare la scelta del drive.

Un tipico comando CP/M è:

```
TYPE KEYS.CCP
```

...dove TYPE è la funzione richiesta che indica di visualizzare sullo schermo e KEYS.CCP è il nome del file che si vuole visualizzare.

Attivazione del drive per difetto.

Se si ha un drive addizionale connesso è possibile attivare quello per difetto scegliendo tra il Drive A e il Drive B scrivendo A: o B: al prompt B> o A>. Tale prompt naturalmente, indica che quello è il drive attuale. Aggiungendo A: o B: come prefisso al nome del file non si modifica il drive per difetto.

Comandi diretti

Vi sono molti comandi diretti che possono essere immessi al prompt A> o B>. Ogni comando può essere abbreviato e sebbene le semplici funzioni descritte sono interne vi sono inoltre molti comandi transienti sofisticati che hanno lo stesso nome.

Comando DIR

DIR elenca la DIRectory del disco. I nomi dei file non vengono visualizzati in base ad un ordine particolare ma la posizione del nome del file indica quella della voce del file nella directory del disco. I caratteri jolly sono permessi. I file fissati con l'attributo SYS non vengono elencati.

DIR	elencherà tutti i file del drive per difetto
DIR B:	elencherà tutti i file del Drive B:
DIR *.BAS	elencherà tutti i file di tipo .BAS
DIR B:*.BAS	elencherà tutti i file di tipo .BAS del Drive B:
DIR PIP.COM	elencherà solo il file PIP.COM (se esiste)

Comando DIRSYS o DIRS

DIRSYS o DIRS elenca solo le voci presenti nella directory con attributo SYS altrimenti viene eseguito come DIR. L'attributo SYS è descritto successivamente.

Comando ERASE o ERA

Il comando ERA è usato per cancellare i file dalla directory. Vengono cancellate solo le voci presenti nella directory, quindi i dati restano nella sezione di disco fino a quando tale spazio non viene riutilizzato da un altro file; l'informazione non è comunque recuperabile. I caratteri jolly sono permessi e se usati ERA chiede conferma. ERA non elenca i nomi dei file che vengono cancellati. Se un file da cancellare è di tipo solo lettura (Read/Only) il comando verrà abortito. L'attributo solo lettura verrà descritto successivamente.

ERA PIP.COM	cancellerà il file PIP.COM
ERA B:PIP.COM	cancellerà il file PIP.COM sul Drive B:
ERA *.BAS	cancellerà tutti i file .BAS

Comando RENAME o REN

REN permette di rinominare un file esistente. Prima di tutto occorre specificare il nuovo nome a cui fa seguito il segno = e successivamente il nome di file esistente. Se il nuovo nome esiste verrà visualizzato un messaggio di errore.

I caratteri jolly non possono essere usati con il comando REN ed è quindi necessario usare il comando RENAME.COM.

REN NUONOME.BAS=VECNOME.BAS rinominerà il file VECNOME.BAS con NUONOME.BAS

REN B:NUONOME.BAS=VECNOME.BAS rinominerà il file VECNOME.BAS con NUONOME.BAS sul Drive B.

Comando TYPE o TYP

TYPE visualizza il file specificato sullo schermo. Se il file non è di tipo ASCII si possono verificare effetti strani e indesiderati.

TYPE KEYS.CCP

...visualizzerà il file KEYS.CCP

Comando USER o USE

USER modifica il numero utente attuale. CP/M inizia con il numero utente attuale fissato a 0. Di solito è possibile accedere solo ai file identificati con numero utente attuale, fornendo così un metodo di suddivisione della directorty.

Ad un file in User 0 con attributo SYS possono accedere tutti i numeri utente. Questa è una caratteristica molto utile poichè rende disponibili a tutti gli utenti i programmi applicativi, senza rendere necessaria una copia in ogni area utente.

USER 3

...fissa il numero utente a 3

Comandi transienti

Per effettuare gestione di file più sofisticata di quella permessa in modo diretto occorre utilizzare uno dei programmi di utilità forniti. Questi sono richiamabili scrivendo il nome del programma a cui fa seguito il nome del file e/o alcuni parametri. Si è già probabilmente usato DISCKIT.

I comandi vengono suddivisi in diverse categorie indicate tra breve.

I comandi DISCKIT ., SETKEYS, SETLST, SETSIO, PALETTE, LANGUAGE e AMSDOS sono progettate da AMSTRAD e sono eseguibili solo su sistemi AMSTRAD. Non sono eseguibili su qualsiasi altro sistema CP/M.

E' possibile inserire in una linea comandi multipli; tali comandi sono separati da un punto esclamativo. Ad esempio:

LANGUAGE 3!SETKEYS KEYS.WP

Gestione delle periferiche

DISCKIT è un programma completo che formatta, copia e controlla. E' più veloce formattare in fase di copia che formattare e poi copiare. Menù molto chiari indicano quali tasti funzione occorra premere. I dischi in vendita sono una forma speciale di dischi di sistema e sono progettati per la distribuzione del software, anche se i dischi di dati sono più indicati per tale attività in ambiente CP/M Plus.

ATTENZIONE

La licenza d'uso del proprio CP/M (determinato elettronicamente mediante numero seriale) ne permette l'uso su un solo sistema. Ciò significa in particolare che è proibito dare ad altre persone un disco di copia del CP/M con il PROPRIO numero seriale. Poichè le copia del lato 1 (Side 1) del pacchetto software originale includeranno il proprio CP/M (nel file .EMS), si deve fare attenzione a non vendere, scambiare o separare un qualsiasi disco con tale file al suo interno.

I caratteri

Il 6128 è dotato di un set di caratteri internazionale. Il comando LANGUAGE scambia alcuni di questi caratteri in modo che anche il software piuttosto semplice possa visualizzare caratteri alternativi e accentati. Ulteriori informazioni sono contenute nella parte 16 del capitolo "Riferimenti".

Il comando

LANGUAGE 3

...fisserà il set di caratteri inglese, il quale scambia i segni # e £ (apparsi nel set per difetto americano).

Colori

I colori per difetto del CP/M Plus sul 6128 (con monitor a colori) sono il bianco brillante su sfondo blu. Questi colori possono essere modificati mediante comando PALETTE, il quale può assumere diversi parametri, uno per ogni inchiostro; ink 0 opera sullo sfondo e il bordo mentre ink 1 opera sul testo. Ogni colore è rappresentato da un numero compreso tra 0 e 63; questo tipo di comportamento può essere usato, su un monitor a fosfori bianco, per regolare l'intensità del colore (luminosità).

E' possibile specificare uno qualsiasi dei numeri di inchiostro, da uno a sedici, sebbene solo i primi due saranno visibili in modalità ad 80 colonne.

Il comando:

PALETTE 63,1

...invertirà i parametri normali di ink 0 e ink 1, fornendo sfondo bianco brillante (63) con testo in blu (1).

Per selezionare i colori (o l'intensità) si consulti la seguente tabella. E' possibile usare sia la notazione esadecimale che quella decimale.

Colore	Esadecimale	Decimale	Colore	Esadecimale	Decimale
Nero	&00	0	Blu pastello	&2B	43
Blu	&02	2	Arancio	&2C	44
Blu brillante	&03	3	Rosa	&2E	46
Rosso	&08	8	Magenta pastello	&2F	47
Magenta	&0A	10	Verde brillante	&30	48
Malva	&0B	11	Verde mare	&32	50
Rosso brillante	&0C	12	Azzurro brillante	&33	51
Porpora	&0E	14	Verde limone	&38	56
Magenta brillante	&0F	15	Verde pastello	&3A	58
Verde	&20	32	Azzurro pastello	&3B	59
Azzurro	&22	34	Giallo brillante	&3C	60
Blu cielo	&23	35	Giallo pastello	&3E	62
Giallo	&28	40	Bianco brillante	&3F	63
Bianco	&2A	42			

La tastiera

I codici generati dalla tastiera possono essere modificati dal comando SETKEYS. Ciò permette di assegnare i codici ai tasti e di espandere i token. I codici attuali devono essere scritti in un file il cui nome deve poi essere specificato nel comando SETKEYS. Il file di comandi può essere creato da un editor di testi, da PIP o da BASIC. Ad esempio:

SETKEYS KEYS.TST

...dove KEYS.TST contiene:

E &8C "DIR ↑ M" token di espansione 12
8 N S C" ↑ H"backspace=**[CONTROL]H,**ASCII08

...ridefiniremo prima di tutto il token di espansione **[CONTROL][ENTER]** (rappresentato da &0C) in modo diventi DIR**[RETURN]** e poi si fa diventare il tasto <- (tasto numero 8) tasto di backspace (<-).

I file standard forniti con il 6128 sono KEYS.CCP per i comandi di editing di CP/M e KEYS.WP da utilizzare per molti word processor.

Stampanti

L'inizializzazione delle stampanti può essere fatta dal comando:

SETLST <nomefile>

...dove <nomefile> contiene la stringa o le stringhe da inviare alla stampante. Così come nel file di comando SETKEYS, i codici di controllo possono essere rappresentati da:

↑ <carattere>

...o...

↑ '<valore carattere>'

...o...

↑ '<codice di controllo>'

...dove i codici di controllo sono ESC, FF ecc, come mostrato nella tabella dei caratteri ASCII nel capitolo "Riferimenti"

Un codice di inizializzazione utile per molte stampanti è il valore 15 che pone la stampante in modalità condensata.

Il comando:

PRINT #8,CHR$(15)

...dovrebbe fissare ciò in BASIC. In CP/M il comando è:

SETLST CONDENSE

...dove il file CONDENSE contiene una delle seguenti linee di testo:

↑ 'SI'
↑ 0
↑ '&F'
↑ '15'

...che sono tutte interpretate come numero decimale 15.

Alcuni programmi applicativi richiedono uno schermo 24x80. Il comando SET24x80 serve a tale scopo.

I comandi:

SET24x80

...o...

SET24x80 ON

...attivano la modalità 24x80 e:

SET24X80 OFF

...la disattivano.

Il normale schermo del 6128 è 24x80 con la linea in fondo riservata per i messaggi di errore. Una disabilitazione della modalità 24x80 sarà solo percettibile se anche la linea di stato sarà disabilitata. Per ulteriori informazioni si consulti la parte 15 del Capitolo 7.

Interfaccia seriale

Vi è inoltre una interfaccia seriale (Input/Output) RS232. La sua esistenza può essere provata scrivendo il comando SETSIO (senza parametri):

SETSIO

...o può essere fissata usando un comando che può includere qualche (o tutti) i seguenti parametri:

SETSIO, RX 1200, TX 75, PARITY NONE, STOP 1, BITS 8,
HANDSHAKE ON, XOFF OFF

...tale comando dovrebbe fissare una nuova configurazione.

I baud rate e gli stati XON/XOFF sono effettuati da uno degli assegnamenti possibili del comando DEVICE. DEVICE si occupa dei dispositivi logici e fisici. I dispositivi logici sono indicati da :. Per esaminare tutti gli attributi di un dispositivo attuale si scriva:

DEVICE

...e gli attributi possono essere cambiati da comandi come:

DEVICE SIO[1200] ...fissa SIO a 1200 baud
DEVICE SIO[XON] ...attiva SIO protocollo XON/XOFF
DEVICE SIO[NOXON] ...disattiva SIO protocollo XON/XOFF

Le connessioni tra i dispositivi logici e quelli fisici possono essere modificati. Di solito: CON: è fissato su CRT (tastiera/schermo, AUX: è fissato su SIO (interfaccia seriale) e LST: è fissato su LPT (interfaccia stampante Centronics). Il comando:

DEVICE LST:=SIO

...invierà l'output della stampante all'interfaccia seriale (se connessa).

Si noti come questo è una redirezione di canale, da non confondere con le caratteristiche di copia di file di PIP. I due comandi GET <nomefile> e PUT <nomefile> ridirigono l'input o l'output della console, e l'output della stampante, indicando loro come usare un file piuttosto che il dispositivo di canale.

PIP

L'utilità PIP (Peripheral Interchange Program) permette di trasferire l'informazione tra il computer e le periferiche.

In generale la forma di tale comando è la seguente:

PIP <destinazione>=<origine>

L'<origine> e la <destinazione> possono essere entrambi nomi di file con caratteri jolly permessi nell'origine, o un dispositivo logico. Si possono usare i seguenti dispositivi logici:

Come origine	Come destinazione
CON: console input	CON: console output
AUX: input ausiliare	AUX: output ausiliare
EOF: fine file	LST: stampante
	PRN: stampante con punti di tabulazione, espansione, numeri di linea e fine pagina.

Esempi di PIP:

PIP B:=A*.COM

...copia tutti i file *.COM dal Drive A: nel Drive B:

PIP KEYBOARD.CPM=KEYS.CCP

...effettua una copia di KEYS.CCP e la chiama KEYBOARD.CPM

PIP CON:=KEYS.CCP

...invia il file KEYS.CCP sullo schermo (effetto simile al comando TYPE KEYS.CCP).

PIP LST:=KEYS.CCP

...invia il file KEYS.CCP alla stampante

PIP TYPEIN.TXT=CON:

...pone l'input da tastiera nel file TYPEIN.TXT

Si noti che quest'ultima operazione è determinata da un codice di controllo **[CONTROL]Z** e che per ottenere una nuova linea occorre scrivere ogni volta **[CONTROL]J** dopo **[RETURN]. [CONTROL]J** in codice ASCII è il simbolo a capo (line feed).

Se scritto senza parametri PIP visualizza il prompt * ed è possibile inserire i comandi che si desiderano. Questa forma è particolarmente utile nella copia di file quando non si ha il file PIP.COM nè sul disco origine nè su quello di destinazione. E' possibile caricare PIP dal lato 1 disco di sistema, togliere il disco ed inserire i dischi che si vogliono usare in fase di copia.

Per uscire dall'ambiente PIP è sufficiente premere **[RETURN]** al prompt *.

Si noti che PIP può essere usato per effettuare copie di file da un disco in un altro su un sistema a singolo drive in quanto avverte quando cambiare il disco. Gli identificatori del drive di origine e di quello di destinazione devono essere diversi.

Gestione di sistema

DIR, ERASE, RENAME e TYPE sono programmi transienti con più caratteristiche delle loro copie interne. Così come per gli altri programmi transienti della Digital Research, i parametri secondari sono specificati tra parentesi quadre. Tutti i dettagli sono contenuti nei file HELP (sul lato 3 dei dischi di sistema). Alcuni esempi sono:

DIR [FULL]	visualizza la dimensione del file e gli attributi
ERASE *.COM [CONFIRM]	richiede la conferma ad ogni file trovato

RENAME	richiederà l'inserimento del nuovo nome e del vecchio
RENAME *.SAV=*.BAK	darà un nome nuovo a tutti i file di tipo .BAK nel tipo .SAV
TYPE KEYS.WP [NOPAGE]	eliminerà l'impaginazione sullo schermo

Gli attributi del file SYS (System) e RO (Read/Only) sono stati già menzionati. Questi e molti altri attributi possono essere assegnati mediante comando SET il quale accetta inoltre l'uso dei caratteri jolly.

I comandi:

SET *.COM [RO]
SET KEYS.CCP [RO]
SET A:[RO]

...fissano i file o il drive nello stato Read/Only, al fine di evitare cancellazioni involontarie.

I comandi:

SET *.COM [RW]
SET KEYS.CCP [RW]
SET A:[RW]

...riportano i file o il drive nello stato Read/Write.

I comandi:

SET *.COM [SYS]
SET KEYS.CCP[SYS]

..forniscono al file l'attributo System (sistema). I file con tali attributi non sono visualizzati dal comando DIR (occorre usare DIRS o DIRSYS). I file sono comunque disponibili e, in oltre, se tali file sono allocati nell'area User numero 0 questi sono a disposizione di tutti gli utenti.

I comandi:

SET *.COM [DIR]
SET KEYS.CCP [DIR]

...eliminano l'attributo System.

Ad ogni disco è possibile assegnare una Label (etichetta) o una Password (parola riservata). La password proteggerà la directory stessa più che i file della directory. E' possibile, inoltre, assegnare una password ai file stessi.

SET [NOME=ORLANDO]
SET [PASSWORD=SILVIO]
SET [PROTECT=ON]

...agiscono sul drive per difetto.

SET *.*[PASSWORD=SILVIO]
SET *.*[PROTECT=READ]

...agiscono sui file del disco per difetto (i caratteri jolly *.* indicano "tutti i file").

La data e l'ora di stampa possono essere attivate dal comando INITDIR (sul lato 2 del disco di sistema). I comandi:

INITDIR
SET [CREATE=ON] ...o... SET [ACCESS=ON] ...e...
SET [UPDATE=ON] ...con...
DIR [FULL]

...inizializzeranno e visualizzeranno la data e l'ora di stampa sul drive per difetto. Scrivendo:

DATE SET

...viene richiesto ad ogni attivazione di CP/M di regolare l'orologio. Una volta fissato, l'orologio terrà traccia dell'ora, e verrà continuamente aggiornato dal 6128 e controllato da:

DATE ...e...
DATE CONTINUOUS

ATTENZIONE

Se è stata attivata la password, l'etichetta su disco, o l'ora e la data, si raccomanda di non far scrivere MAI tale disco da AMSDOS o CP/M 2.2, nè da alcun dispositivo che abbia tali caratteristiche.

Di solito si accederà solo ai file del drive per difetto a meno che non ne venga specificato uno particolare. Il comando:

SETDEF *,A:

...(dove * indica il drive per difetto) indica a CP/M (quando ricerca i file) di cercare prima nel drive per difetto e poi nel Drive A:. In altre parole, se il drive per difetto è B: i file verranno cercati automaticamente lì anche se esistono sono nel Drive A:.

I comandi:

SETDEF [PAGE] ...e...
SETDEF [NOPAGE]

...attivano e disattivano l'impaginazione automatica sullo schermo.

Si ricorda che molte delle caratteristiche DEVICE, SET e SETDEF in particolare quando si riferiscono ai due drive (piuttosto che ai file o ai dischi) necessitano, ogni volte si esegue CP/M, di essere fissate, oltre alla data. Questa è la principale applicazione del file PROFILE.SUB.

Per eseguire file o comandi singoli automaticamente occorre usare SUBMIT. I contenuti dei file di comando sono di tipo testo ed è possibile includerli nei programmi se il primo carattere delle linee del file .SUB è <.

La dimensione del drive, l'ammontare di spazio e il numero di voci della directory di un disco oltre alle aree utenti contenenti file e l'etichetta del disco possono essere visualizzate mediante combinazioni del comando SHOW:

SHOW B:
SHOW B:[LABEL]
SHOW B:[USERS]
SHOW B:[DIR]
SHOW B.[DRIVE]

...richiedono tutte informazioni sul drive B:

Procedura di uscita di CP/M Plus

AMSDOS

Questo programma abbandona il controllo da CP/M e torna nel BASIC AMSTRAD; da quel punto in poi sono disponibili i comandi di disco di AMSDOS.

Capitolo 6
Riferimenti

Questo capitolo fornisce la maggior parte delle informazioni di riferimento che è probabile occorreranno imparando ad utilizzare questo computer.

Argomenti esaminati:

* Posizione del cursore ed estensione dei codici di controllo
* Interruzioni (Interrupts)
* Caratteri ASCII e grafici
* Riferimenti ai tasti
* Suono
* Messaggi di errore
* Parole chiave del BASIC
* Tabelle
* Collegamenti
* Stampanti
* Joystick
* Organizzazione dei dischi
* RSX (Resident System eXtensions)
* Memoria
* Emulatore di terminale di CP/M Plus
* Set di caratteri di CP/M Plus

Parte 1: Posizione del cursore ed estensione dei codici di controllo mediante BASIC

In molti programmi applicativi, il cursore del testo può essere posizionato all'esterno della finestra attuale. Varie operazioni costringeranno il cursore ad una posizione lecita:

1. Scrivere il carattere

2. Disegnare il cursore

3. Eseguire i codici di controllo segnati da un asterisco nella lista che segue.

La procedura che porta il cursore in una posizione lecita è la seguente:

1. Se il cursore è a destra del margine destro, viene portato alla colonna più a sinistra della linea seguente.

2. Se il cursore è a sinistra del margine sinistro, viene portato alla colonna più a destra della linea precedente.

3. Se il cursore è sopra il margine superiore, la finestra viene fatta scorrere di una linea ed il cursore viene portato sulla riga più in alto nella finestra.

4. Se il cursore è sopra il margine inferiore, la finestra viene fatta scorrere di una linea ed il cursore viene portato sulla riga più in basso nella finestra.

I controlli e le operazioni vengono effettuate esattamente nell'ordine riportato. Le posizioni illegali del cursore possono essere uguali a zero o negative che si trovano a sinistra o al di sopra dello schermo.

I caratteri tra 0 e 31 inviati allo schermo non producono un carattere ma vengono interpretati come codici di controllo (e come tali non devono essere impiegati a sproposito). Alcuni codici modificano il significato dei caratteri seguenti che costituiscono i parametri del codice.

Un codice di controllo inviato allo schermo grafico stamperà semplicemente il simbolo collegato alla sua funzione se acceduta tramite tastiera (ad esempio &07 "BEL" - **[CTRL] G**). Eseguirà invece la sua funzione di controllo se indirizzato usando il comando:

PRINT CHR$(&07) o PRINT" ⍾ "(dove il simbolo ⍾ è ottenuto premendo **[CTRL] G** all'interno dell'istruzione PRINT.

I codici indicati da un asterisco (*) costringono il cursore ad una posizione lecita nella finestra corrente prima di essere eseguiti ma possono lasciare il cursore in una posizione non lecita. I codici ed il loro significato sono descritti prima dal loro valore in esadecimale (&XX) e poi dal corrispondente decimale.

Caratteri di controllo BASIC

Valore	Nome	Parametro	Significato
&00 0	NUL		Nessun effetto. Ignorato
&01 1	SOH	da 0 a 255	Stampa il simbolo indicato dal parametro. Permette di visualizzare i simboli tra 0 e 31.
&02 2	STX		Elimina il cursore del testo. Equivale al comando CURSOR con parametro = 0.
&03 3	ETX		Riattiva il cursore del testo. Equivale al comando CURSOR con parametro = 1. Notare che per visualizzare un cursore da un programma BASIC (che non sia il cursore automatico generato quando il BASIC attende un input da tastiera), deve essere usato il comando CURSOR con parametro 1.
&04 4	EOT	da 0 a 2	Fissa la modalità di schermo. Parametro MOD 4. Equivalente al comando MODE.
&05 5	ENQ	da 0 a 255	Invia il carattere indicato dal parametro allo schermo grafico.
&06 6	ACK		Abilita lo schermo di testo (Vedere &015 NAK)
&07 7	BEL		Emette un Bip. Svuota le code relative ai suoni.
&08 8	* BS		Sposta il cursore indietro di un carattere.

Valore	Nome	Parametro	Significato
&09 9	* TAB		Sposta il cursore avanti di un carattere.
&0A 10	* LF		Sposta il cursore verso il basso di una linea.
&0B 11	* VT		Sposta il cursore verso l'alto di una linea.
&0C 12	FF		Cancella la finestra di testo e porta il cursore nell'angolo in alto a sinistra. Equivale al comando CLS.
&0D 13	* CR		Porta il cursore sul primo carattere a sinistra della stessa linea.
&0E 14	SO	da 0 a 15	Fissa il colore della carta. Parametro MOD 16. Equivale al comando PAPER.
&0F 15	SI	da 0 a 15	Fissa il colore della penna. Parametro MOD 16. Equivale al comando PEN.
&10 16	* DLE		Cancella il carattere corrente. Riempie la griglia che costituisce il carattere con il colore della carta.
&11 17	* DC1		Cancella la linea dal margine sinistro della finestra fino alla posizione (inclusa) del cursore. Riempie le posizioni interessate con il colore della carta.
&12 18	* DC2		Cancella la linea dalla posizione (inclusa) del cursore al margine destro della finestra. Riempie le posizioni interessate con il colore della carta.
&13 19	* DC3		Cancella il contenuto della finestra fino alla posizione del cursore. Riempie le posizioni interessate con il colore della carta.

Valore	Nome	Parametro	Significato
&14 20	* DC4		Cancella il contenuto della finestra dalla posizione del cursore in poi. Riempie le posizioni interessate con il colore della carta.
&15 21	NAK		Disattiva lo schermo di testo. Lo schermo non reagirà ad alcun comando ad esso inviato fino alla ricezione di un carattere ACK (&06 6).
&16 22	SYN	da 0 a 1	Parametro MOD 2. Opzione Trasparenza. 0 disabilita 1 abilita.
&17 23	ETB	da 0 a 3	Parametro MOD 4. 0 fissa il normale modo grafico 1 fissa il modo grafico XOR 2 fissa il modo grafico AND 3 fissa il modo grafico OR
&18 24	CAN		Scambia i colori di carta e penna.
&19 25	EM	da 0 a 255 da 0 a 255 da 0 a 255 da 0 a 255 da 0 a 255 da 0 a 255 da 0 a 255 da 0 a 255 da 0 a 255	Riempie la matrice di un carattere definibile dall'utente. Equivale al comando SYMBOL. Prende 9 parametri. Il primo indica il carattere da riempire. Gli altri 8 specificano la matrice. Il bit più significativo del primo byte corrisponde al pixel in alto a sinistra della cella del carattere, il bit meno significativo dell'ultimo byte corrisponde al pixel in basso a destra della cella del carattere.
&1A 26	SUB	da 1 a 80 da 1 a 80 da 1 a 25 da 1 a 25	Fissa la finestra. Equivale al comando WINDOW. I primi due parametri specificano i margini sinistro e destro della finestra (il valore più piccolo viene considerato il margine sinistro, il più grande il margine destro). Gli altri due parametri specificano i margini superiore ed inferiore della finestra (il valore più piccolo viene considerato il margine superiore, il più grande il margine inferiore).

Valore	Nome	Parametro	Significato
&1B 27	ESC		Nessun effetto. Ignorato.
&1C 28	FS	da 0 a 15 da 0 a 31 da 0 a 31	Fissa l'inchiostro ad una coppia di colori. Equivale al comando INK. Il primo parametro (MOD 16) specifica l'inchiostro, gli altri due (MOD 32) i colori richiesti. I valori compresi tra 27 e 31 corrispondono a colori non definiti.
&1D 29	GS	da 0 a 31 da 0 a 31	Fissa il bordo ad una coppia di colori. Equivale a comand BORDER. I due parametri (MOD 32) specificano i due colori. I valor compresi tra 27 e 31 corrispondono a colori non definiti.
&1E 30	RS		Porta il cursore nella posizione in alto a sinistra della finestra.
&1F 31	US	da 0 a 80 da 0 a 25	Porta il cursore nella posizione indicata nella finestra corrente. Equivale al comando LOCATE. Il primo parametro costituisce la colonna su cui portarsi, il secondo corrisponde alla linea.

La gestione del 464/6128 è affidata ad un sofisticato sistema operativo operante in tempo reale. Il sistema operativo "dirige il traffico" tra input e output.

Innanzitutto fornisce l'interfaccia tra l'hardware e l'interprete BASIC (ad esempio nel caso del colore lampeggiante, dove il BASIC passa semplicemente dei parametri ed il sistema operativo esegue l'ordine; in tal modo una parte decide cosa deve essere fatto e l'altra come ciò deve essere fatto.

Il sistema operativo della macchina è conosciuto generalmente come il "firmware" ed è composto dalle routine in codice macchina chiamate dai comandi ad alto livello del BASIC.

Se si è tentati di fare delle POKE negli indirizzi della memoria della macchina o effettuare delle CALL alle subroutine, è conveniente salvare prima il programma contenuto in memoria; dopo tali operazioni è possibile perderlo.

Se si intende utilizzare molto il linguaggio macchina, sarà necessario utilizzare un assemblatore (assembler).

Parte 2: Interrupts (Interruzioni)

Il 464/6128 fa un grande uso degli interrupt dello Z80 e fornisce un sistema operativo che include molte possibilità di multi-tasking, esemplificate dalle strutture AFTER e EVERY descritte precedentemente in questo manuale. La precedenza degli eventi è:

Break (**[ESC][ESC]**)
Timer 3
Timer 2 (e le tre code del suono)
Timer 1
Timer 0

Le interruzioni dovrebbero essere incluse solo dopo aver considerato le conseguenze dei possibili stati intermedi delle variabili al momento dell'interruzione. La stessa sub-routine di interruzione dovrebbe evitare interazioni indesiderate con lo stato delle variabili del programma principale.

Le code del suono hanno interruzioni indipendenti di uguale priorità. Quando inizia una interruzione del suono non può essere interrotta da nessun'altra interruzione del suono. Ciò permette di far condividere le variabili alle routine di interruzione del suono senza che sorga alcun problema.

Quando l'interruzione di una coda del suono è abilitata (usando ON SQ GOSUB), agirà immediatamente se la coda del suono di quel canale non è piena, altrimenti agirà quando termina il suono attualmente prodotto e si crea uno spazio nella coda. L'interruzione disabiliterà l'evento, in modo che la sub-routine possa riabilitarsi se sono richieste altre interruzioni.

Cercando di produrre un suono o controllando lo stato della coda si disabilita una interruzione del suono.

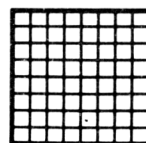
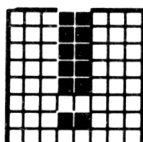
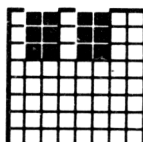
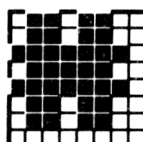
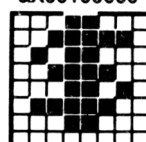
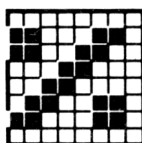
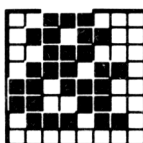
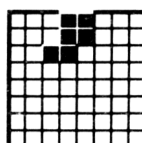
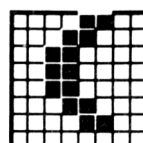
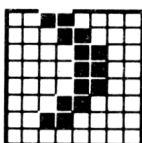
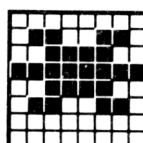
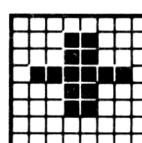
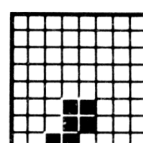
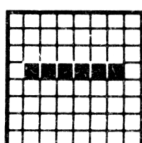
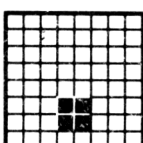
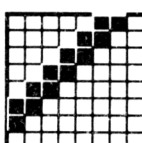
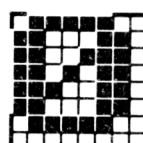
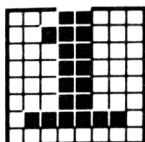
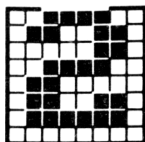
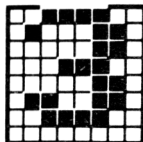
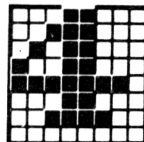
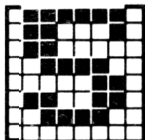
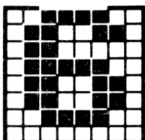
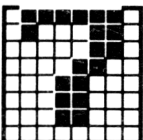
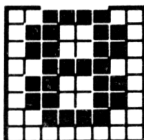
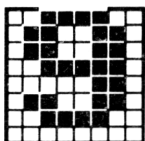
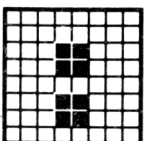
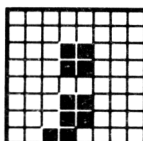
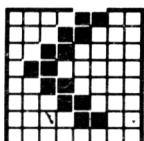
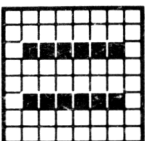
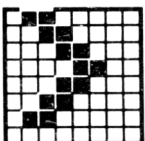
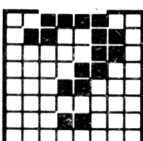
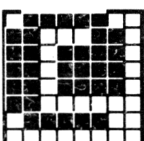
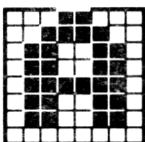
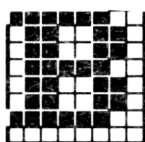
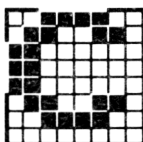
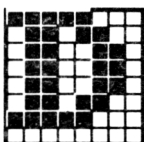
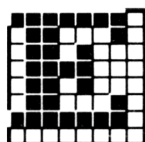
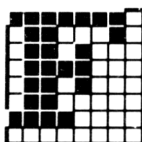
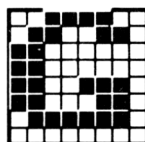
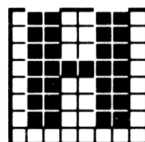
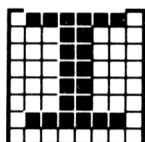
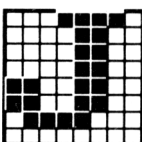
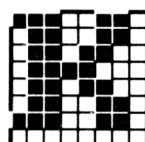
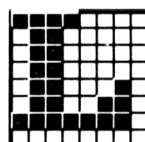
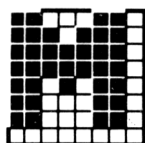
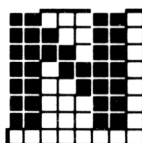
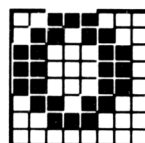
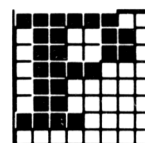
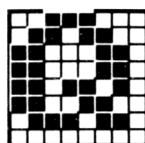
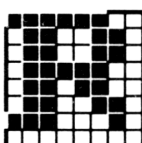
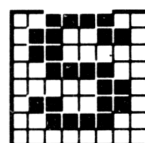
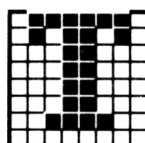
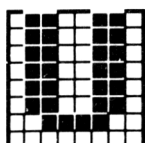
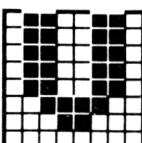
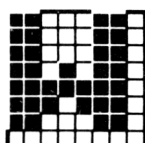
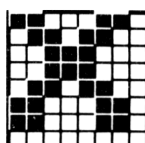
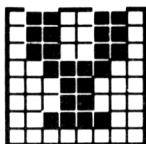
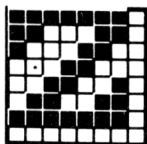
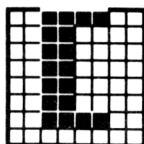
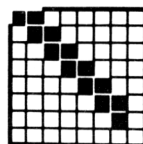
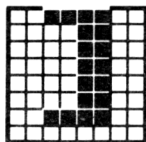
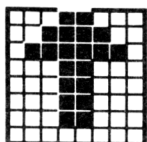
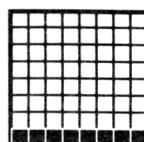
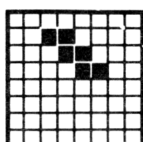
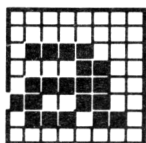
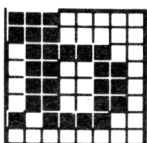
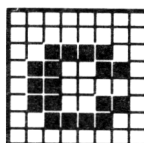
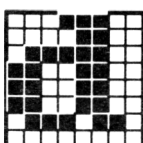
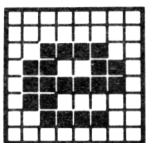
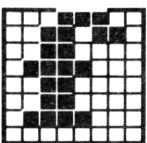
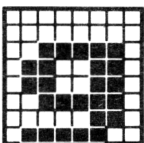
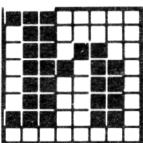
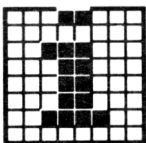
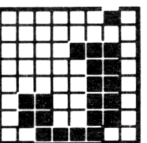
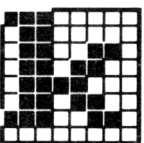
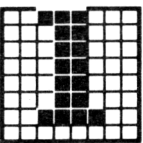
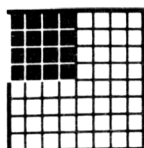
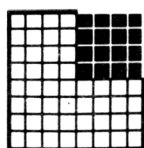
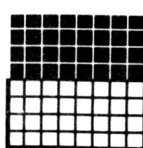
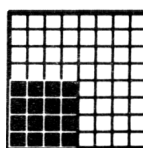
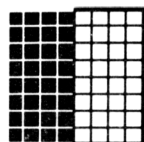
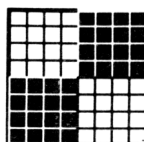
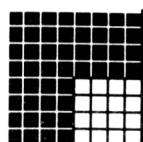
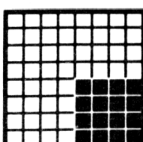
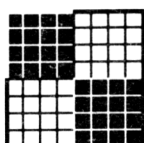
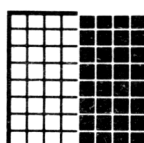
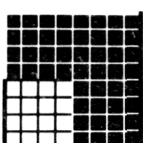
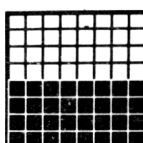
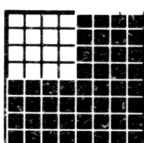
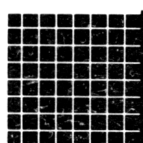
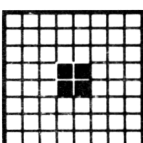
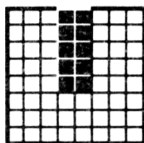
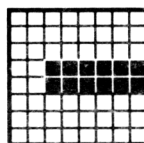
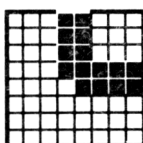
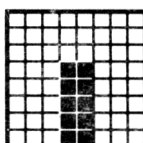
Parte 3: Caratteri ASCII e grafici del BASIC

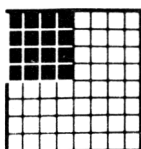
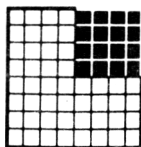
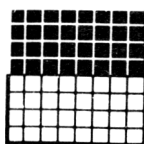
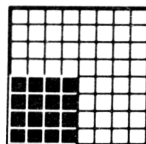
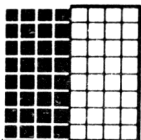
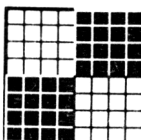
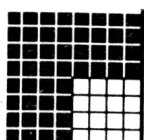
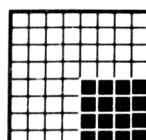
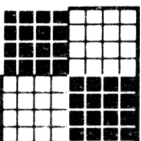
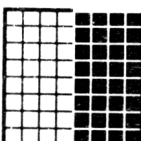
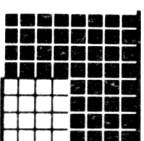
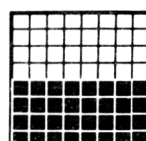
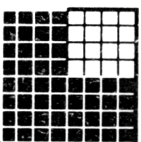
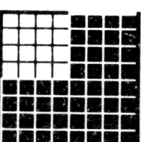
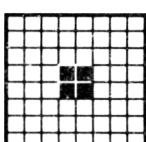
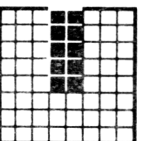
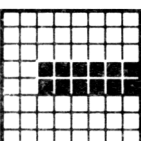
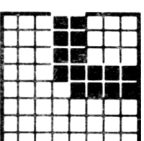
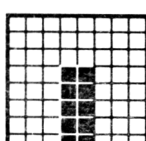
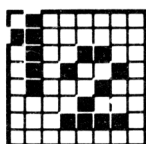
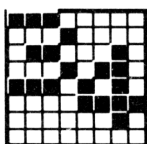
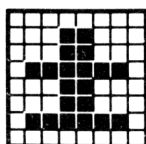
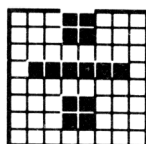
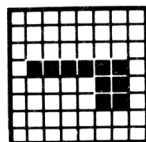
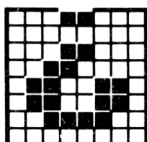
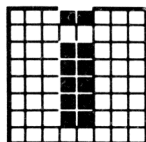
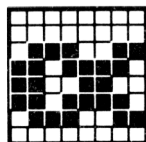
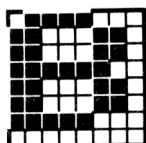
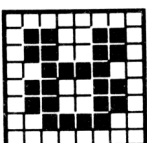
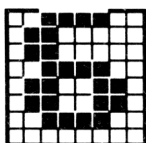
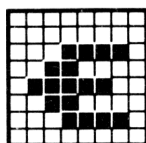
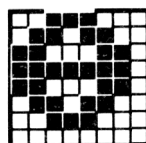
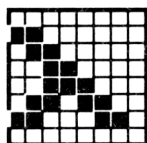
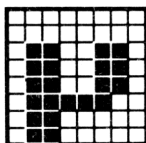
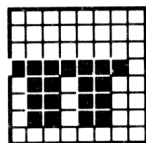
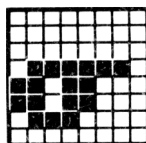
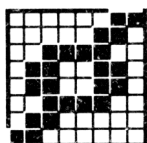
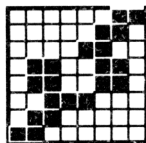
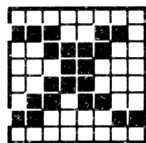
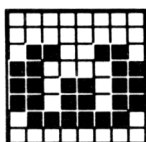
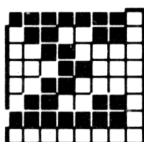
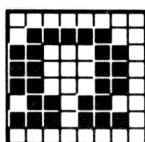
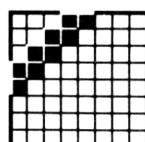
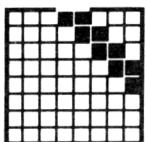
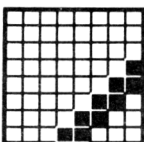
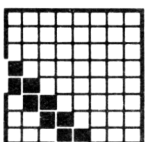
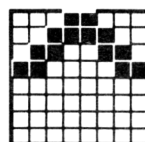
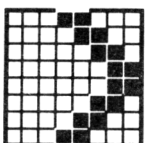
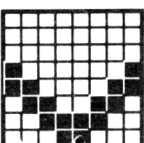
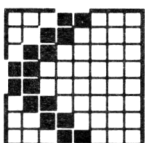
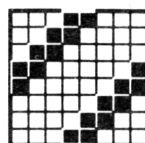
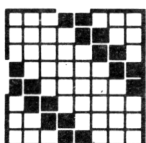
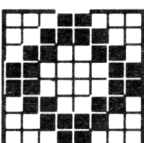
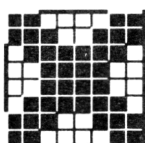
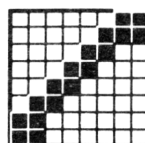
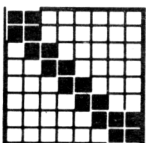
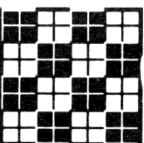
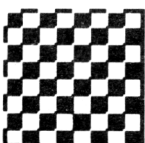
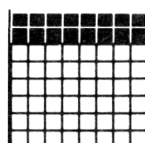
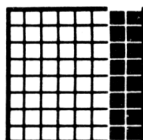
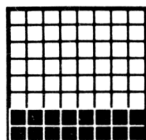
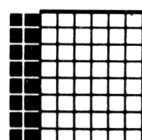
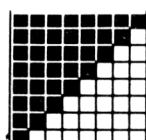
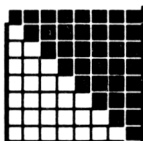
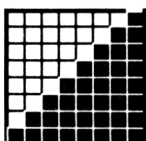
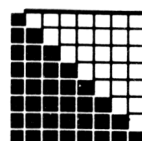
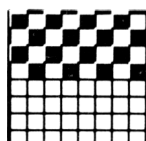
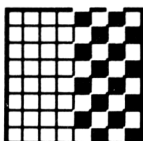
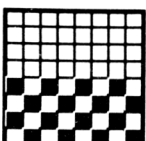
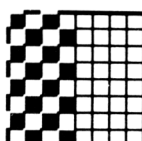
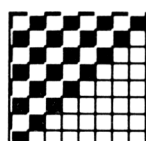
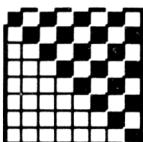
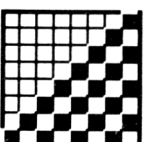
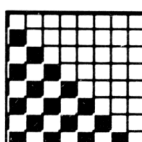
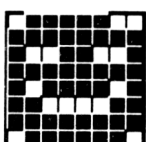
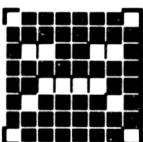
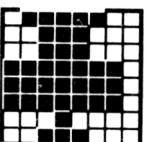
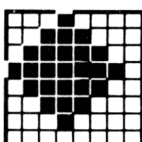
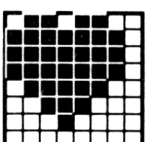
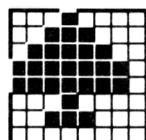
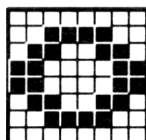
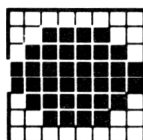
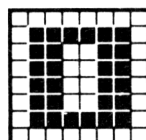
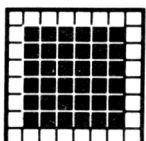
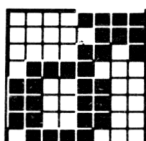
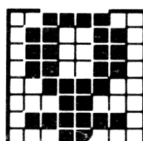
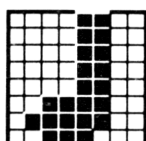
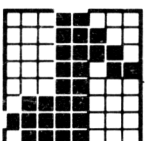
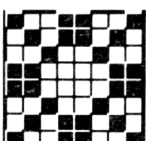
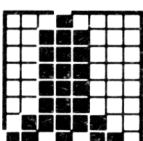
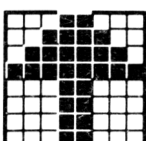
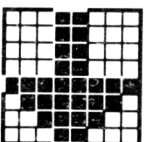
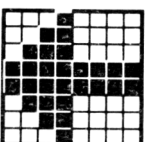
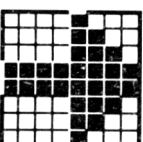
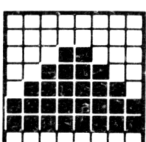
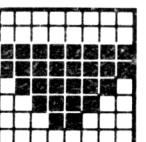
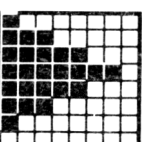
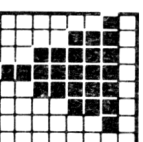
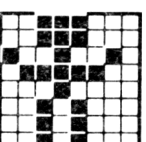
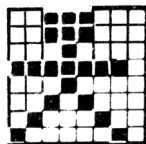
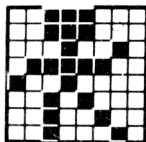
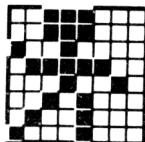
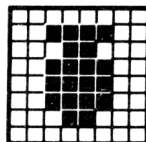
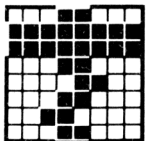
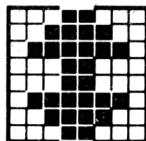
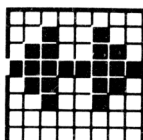
ASCII

La tabella riportata illustra il normale insieme di caratteri ASCII usando la notazione decimale, ottale ed esadecimale, oltre ai codici ASCII dove sia il caso. Ognuna delle celle dei caratteri del 464/6128 viene rappresentata in dettaglio nelle pagine seguenti.

DEC	OTT	ESA	Caratteri ASCII	DEC	OTT	ESA	ASCII	DEC	OTT	ESA	ASCII
0	000	00	NUL ([CTRL]@)	50	062	32	2	100	144	64	d
1	001	01	SOH ([CTRL]A)	51	063	33	3	101	145	65	e
2	002	02	STX ([CTRL]B)	52	064	34	4	102	146	66	f
3	003	03	ETX ([CTRL]C)	53	065	35	5	103	147	67	g
4	004	04	EOT ([CTRL]D)	54	066	36	6	104	150	68	h
5	005	05	ENQ ([CTRL]E)	55	067	37	7	105	151	69	i
6	006	06	ACK ([CTRL]F)	56	070	38	8	106	152	6A	j
7	007	07	BEL ([CTRL]G)	57	071	39	9	107	153	6B	k
8	010	08	BS ([CTRL]H)	58	072	3A	:	108	154	6C	l
9	011	09	HT ([CTRL]I)	59	073	3B	;	109	155	6D	m
10	012	OA	LF ([CTRL]J)	60	074	3C	<	110	156	6E	n
11	013	OB	VT ([CTRL]K)	61	075	3D	=	111	157	6F	o
12	014	OC	FF ([CTRL]L)	62	076	3E	>	112	160	70	p
13	015	OD	CR ([CTRL]M)	63	077	3F	?	113	161	71	q
14	016	OE	SO ([CTRL]N)	64	100	40	@	114	162	72	r
15	017	OF	SI ([CTRL]O)	65	101	41	A	115	163	73	s
16	020	10	DLE ([CTRL]P)	66	102	42	B	116	164	74	t
17	021	11	DC1 ([CTRL]Q)	67	103	43	C	117	165	75	u
18	022	12	DC2 ([CTRL]R)	68	104	44	D	118	166	76	v
19	023	13	DC3 ([CTRL]S)	69	105	45	E	119	167	77	w
20	024	14	DC4 ([CTRL]T)	70	106	46	F	120	170	78	x
21	025	15	NAK ([CTRL]U)	71	107	47	G	121	171	79	y
22	026	16	SYN ([CTRL]V)	72	110	48	H	122	172	7A	z
23	027	17	ETB ([CTRL]W)	73	111	49	I	123	173	7B	{
24	030	18	CAN ([CTRL]X)	74	112	4A	J	124	174	7C	|
25	031	19	EM ([CTRL]Y)	75	113	4B	K	125	175	7D	}
26	032	1A	SUB ([CTRL]Z)	76	114	4C	L	126	176	7E	~
27	033	1B	ESC	77	115	4D	M				
28	034	1C	FS	78	116	4E	N				
29	035	1D	GS	79	117	4F	O				
30	036	1E	RS	80	120	50	P				
31	037	1F	US	81	121	51	Q				
32	040	20	SP	82	122	52	R				
33	041	21	!	83	123	53	S				
34	042	22	"	84	124	54	T				
35	043	23	#	85	125	55	U				
36	044	24	$	86	126	56	V				
37	045	25	%	87	127	57	W				
38	046	26	&	88	130	58	X				
39	047	27	'	89	131	59	Y				
40	050	28	(	90	132	5A	Z				
41	051	29	)	91	133	5B	[				
42	052	2A	*	92	134	5C	\				
43	053	2B	+	93	135	5D	]				
44	054	2C	,	94	136	5E	↑				
45	055	2D	-	95	137	5F	_				
46	056	2E	.	96	140	60	`				
47	057	2F	/	97	141	61	a				
48	060	30	Ø	98	142	62	b				
49	061	31	1	99	143	63	c				

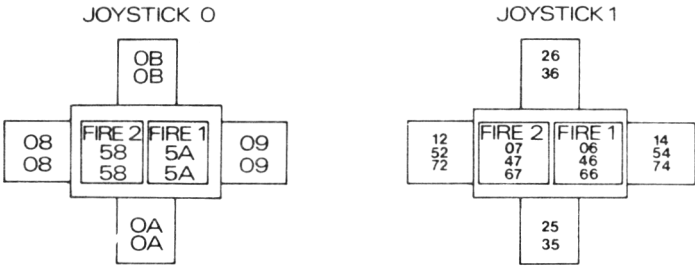
Set di caratteri grafici BASIC specifici per la macchina

I caratteri qui riprodotti vengono riportati su una normale matrice 8 x 8 usata per la scrittura sullo schermo del 464/6128. I caratteri definibili dall'utente possono essere riuniti per ottenere speciali effetti, ponendoli uno in fianco all'altro. Vedere la parte "Caratteri definibili dall'utente" nel Capitolo "A vostra disposizione ...".

49
&H31
&X00110001

50
&H32
&X00110010

51
&H33
&X00110011

52
&H34
&X00110100

53
&H35
&X00110101

54
&H36
&X00110110

55
&H37
&X00110111

56
&H38
&X00111000

57
&H39
&X00111001

58
&H3A
&X00111010

59
&H3B
&X00111011

60
&H3C
&X00111100

61
&H3D
&X00111101

62
&H3E
&X00111110

63
&H3F
&X00111111

64
&H40
&X01000000

65
&H41
&X01000001

66
&H42
&X01000010

67
&H43
&X01000011

68
&H44
&X01000100

69
&H45
&X01000101
70
&H46
&X01000110
71
&H47
&X01000111
72
&H48
&X01001000
73
&H49
&X01001001
74
&H4A
&X01001010
75
&H4B
&X01001011
76
&H4C
&X01001100
77
&H4D
&X01001101
78
&H4E
&X01001110
79
&H4F
&X01001111
80
&H50
&X01010000
81
&H51
&X01010001
82
&H52
&X01010010
83
&H53
&X01010011
84
&H54
&X01010100
85
&H55
&X01010101
86
&H56
&X01010110
87
&H57
&X01010111
88
&H58
&X01011000

89
&H59
&X01011001
90
&H5A
&X01011010
91
&H5B
&X01011011
92
&H5C
&X01011100
93
&H5D
&X01011101
94
&H5E
&X01011110
95
&H5F
&X01011111
96
&H60
&X01100000
97
&H61
&X01100001
98
&H62
&X01100010
99
&H63
&X01100011
100
&H64
&X01100100
101
&H65
&X01100101
102
&H66
&X01100110
103
&H67
&X01100111
104
&H68
&X01101000
105
&H69
&X01101001
106
&H6A
&X01101010
107
&H6B
&X01101011
108
&H6C
&X01101100

109
&H6D
&X01101101
110
&H6E
&X01101110
111
&H6F
&X01101111
112
&H70
&X01110000
113
&H71
&X01110001
114
&H72
&X01110010
115
&H73
&X01110011
116
&H74
&X01110100
117
&H75
&X01110101
118
&H76
&X01110110
119
&H77
&X01110111
120
&H78
&X01111000
121
&H79
&X01111001
122
&H7A
&X01111010
123
&H7B
&X01111011
124
&H7C
&X01111100
125
&H7D
&X01111101
126
&H7E
&X01111110
127
&H7F
&X01111111
128
&H80
&X10000000

129
&H81
&X10000001
130
&H82
&X10000010
131
&H83
&X10000011
132
&H84
&X10000100
133
&H85
&X10000101
134
&H86
&X10000110
135
&H87
&X10000111
136
&H88
&X10001000
137
&H89
&X10001001
138
&H8A
&X10001010
139
&H8B
&X10001011
140
&H8C
&X10001100
141
&H8D
&X10001101
142
&H8E
&X10001110
143
&H8F
&X10001111
144
&H90
&X10010000
145
&H91
&X10010001
146
&H92
&X10010010
147
&H93
&X10010011
148
&H94
&X10010100

129
&H81
&X10000001
130
&H82
&X10000010
131
&H83
&X10000011
132
&H84
&X10000100
133
&H85
&X10000101
134
&H86
&X10000110
135
&H87
&X10000111
136
&H88
&X10001000
137
&H89
&X10001001
138
&H8A
&X10001010
139
&H8B
&X10001011
140
&H8C
&X10001100
141
&H8D
&X10001101
142
&H8E
&X10001110
143
&H8F
&X10001111
144
&H90
&X10010000
145
&H91
&X10010001
146
&H92
&X10010010
147
&H93
&X10010011
148
&H94
&X10010100

169
&HA9
&X10101001
170
&HAA
&X10101010
171
&HAB
&X10101011
172
&HAC
&X10101100
173
&HAD
&X10101101
174
&HAE
&X10101110
175
&HAF
&X10101111
176
&HB0
&X10110000
177
&HB1
&X10110001
178
&HB2
&X10110010
179
&HB3
&X10110011
180
&HB4
&X10110100
181
&HB5
&X10110101
182
&HB6
&X10110110
183
&HB7
&X10110111
184
&HB8
&X10111000
185
&HB9
&X10111001
186
&HBA
&X10111010
187
&HBB
&X10111011
188
&HBC
&X10111100

189
&HBD
&X10111101
190
&HBE
&X10111110
191
&HBF
&X10111111
192
&HC0
&X11000000
193
&HC1
&X11000001
194
&HC2
&X11000010
195
&HC3
&X11000011
196
&HC4
&X11000100
197
&HC5
&X11000101
198
&HC6
&X11000110
199
&HC7
&X11000111
200
&HC8
&X11001000
201
&HC9
&X11001001
202
&HCA
&X11001010
203
&HCB
&X11001011
204
&HCC
&X11001100
205
&HCD
&X11001101
206
&HCE
&X11001110
207
&HCF
&X11001111
208
&HD0
&X11010000

209
&HD1
&X11010001

210
&HD2
&X11010010

211
&HD3
&X11010011

212
&HD4
&X11010100

213
&HD5
&X11010101

214
&HD6
&X11010110

215
&HD7
&X11010111

216
&HD8
&X11011000

217
&HD9
&X11011001

218
&HDA
&X11011010

219
&HDB
&X11011011

220
&HDC
&X11011100

221
&HDD
&X11011101

222
&HDE
&X11011110

223
&HDF
&X11011111

224
&HE0
&X11100000

225
&HE1
&X11100001

226
&HE2
&X11100010

227
&HE3
&X11100011

228
&HE4
&X11100100

229
&HE5
&X11100101
230
&HE6
&X11100110
231
&HE7
&X11100111
232
&HE8
&X11101000
233
&HE9
&X11101001
234
&HEA
&X11101010
235
&HEB
&X11101011
236
&HEC
&X11101100
237
&HED
&X11101101
238
&HEE
&X11101110
239
&HEF
&X11101111
240
&HF0
&X11110000
241
&HF1
&X11110001
242
&HF2
&X11110010
243
&HF3
&X11110011
244
&HF4
&X11110100
245
&HF5
&X11110101
246
&HF6
&X11110110
247
&HF7
&X11110111
248
&HF8
&X11111000

249
&HF9
&X11111001

250
&HFA
&X11111010

251
&HFB
&X11111011

252
&HFC
&X11111100

253
&HFD
&X11111101

254
&HFE
&X11111110

255
&HFF
&X11111111

Parte 4: Riferimenti ai tasti

Normali valori ASCII (esadecimali)

N/A	21 31	7E 22 32	23 33	24 34	25 35	26 36	27 37	28 38	29 39	1F 5F 30	3D 2D	1E A3 5E	10 10 10	7F 7F 7F	N/A	N/A	N/A
E1 09 09	11 51 71	17 57 77	05 45 65	12 52 72	14 54 74	19 59 79	15 55 75	09 49 69	0F 4F 6F	10 50 70	00 7C 40	1B 7B 5B	0D 0D 0D	N/A	N/A	N/A	
N/A	01 41 61	13 53 73	04 44 64	06 46 66	07 47 67	08 48 68	0A 4A 6A	0B 4B 6B	0C 4C 6C	2A 3A	2B 3B	1D 7D 5D	N/A	N/A	N/A		
N/A	1A 5A 7A	18 58 78	03 43 63	16 56 76	02 42 62	0E 4E 6E	0D 4D 6D	3C 2C	3E 2E	3F 2F	1C 60 5C	N/A	N/A	F8 F4 F0	N/A		
N/A	E0 E0 E0	20 20	N/A	FA F6 F2	F9 F5 F1	FB F7 F3											

Nota: N/A indica un tasto senza un corrispondente valore ASCII

Caratteri di espansione, normali locazioni e valori

N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A | 135 135 135 | 136 136 136 | 137 137 137

N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A | 132 132 132 | 133 133 133 | 134 134 134

N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A | 129 129 129 | 130 130 130 | 131 131 131

N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A N/A | 128 128 128 | N/A | 138 138 138

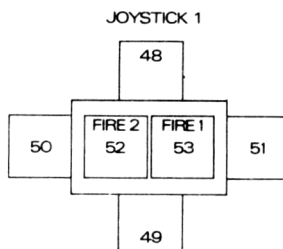
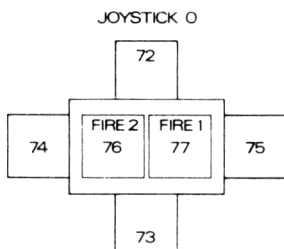
N/A N/A N/A 140 139 139 | N/A | N/A | N/A

Nota: N/A indica un tasto senza un corrispondente valore ASCII

ESPANSIONE CARATTERE	VALORI NORMALI	
	CARATTERE	VALORE ASCII
0 (128)	Ø	&30
1 (129)	1	&31
2 (130)	2	&32
3 (131)	3	&33
4 (132)	4	&34
5 (133)	5	&35
6 (134)	6	&36
7 (135)	7	&37
8 (136)	8	&38
9 (137)	9	&39
10 (138)	.	&2E
11 (139)	[RETURN]	&0D
12 (140)	RUN"[RETURN]	&52 &55 &4E &22 &0D

Nota: Le espansioni dei caratteri da 13 a 31 (da 141 a 159) hanno normalmente valore nullo. Essi hanno i valori assegnati usando il comando KEY di BASIC e vengono assegnati ai tasti usando il comando KEY DEF.

Tasti e numeri dei joystick

66 64 65 57 56 49 48 41 40 33 32 25 24 16 79 10 11 3
68 67 59 58 50 51 43 42 35 34 27 26 17 18 20 12 4
70 69 60 61 53 52 44 45 37 36 29 28 19 13 14 5
21 71 63 62 55 54 46 38 39 31 30 22 21 15 0 7
23 9 47 6 8 2 1

Parte 5: Il suono

Periodi di note e toni

La tabella che segue, fornisce i periodi consigliati per le note nella normale scala temperata, per tutte le otto ottave.

La frequenza prodotta non è esattamente la frequenza richiesta in quanto il periodo deve essere composto da un intero. L'ERRORE RELATIVO è il rapporto percentuale della differenza tra la frequenza corretta e quella effettiva.

NOTA	FREQUENZA	PERIODO	ERRORE RELATIVO	
Do	16.352	3822	-0.007%	
Do #	17.324	3608	+0.007%	
Re	18.354	3405	-0.007%	
Re #	19.445	3214	-0.004%	
Mi	20.602	3034	+0.009%	Ottava -4
Fa	21.827	2863	-0.016%	
Fa #	23.125	2703	+0.009%	
Sol	24.500	2551	-0.002%	
Sol #	25.957	2408	+0.005%	
La	27,500	2273	+0.012%	
La #	29.135	2145	-0.008%	
Si	30.868	2025	+0.011%	

NOTA	FREQUENZA	PERIODO	ERRORE RELATIVO	
Do	32.703	1911	-0.007%	
Do #	34.648	1804	+0.007%	
Re	36.708	1703	+0.022%	
Re #	38.891	1607	-0.004%	
Mi	41.203	1517	+0.009%	
Fa	43.654	1432	+0.019%	Ottava 3
Fa #	46.249	1351	-0.028%	
Sol	48.999	1276	+0.037%	
Sol #	51.913	1204	+0.005%	
La	55.000	1136	-0.032%	
La #	58.270	1073	+0.039%	
Si	61.735	1012	-0.038%	

NOTA	FREQUENZA	PERIODO	ERRORE RELATIVO	
Do	65.406	956	+0.046%	
Do #	69.296	902	+0.007%	
Re	73.416	851	-0.037%	
Re #	77.782	804	+0.058%	
Mi	82.407	758	-0.057%	
Fa	87.307	716	+0.019%	Ottava 2
Fa #	92.499	676	+0.046%	
Sol	97.999	638	+0.037%	
Sol #	103.826	602	+0.005%	
La	110.000	568	-0.032%	
La #	116.541	536	-0.055%	
Si	123.471	506	-0.038%	

NOTA	FREQUENZA	PERIODO	ERRORE RELATIVO	
Do	130.813	478	+0.046%	
Do #	138.591	451	+0.007%	
Re	146.832	426	+0.081%	
Re #	155.564	402	+0.058%	
Mi	164.814	379	-0.057%	
Fa	174.614	358	+0.019%	Ottava-1
Fa #	184.997	338	+0.046%	
Sol	195.998	319	+0.037%	
Sol #	207.652	301	+0.005%	
La	220.000	284	-0.032%	
La #	233.082	268	-0.055%	
Si	246.942	253	-0.038%	

NOTA	FREQUENZA	PERIODO	ERRORE RELATIVO	
Do	261.626	239	+0.046%	Do centrale
Do #	277.183	225	-0.215%	
Re	293.665	213	+0.081%	
Re #	311.127	201	+0.058%	
Mi	329.628	190	+0.206%	
Fa	349.228	179	+0.019%	Ottava 0
Fa #	369.994	169	+0.046%	
Sol	391.995	159	-0.277%	
Sol #	415.305	150	-0.328%	
La	440.000	142	-0.032%	La internazionale
La #	466.164	134	-0.055%	
Si	493.883	127	+0.356%	

NOTA	FREQUENZA	PERIODO	ERRORE RELATIVO
Do	523.251	119	-0.374%
Do #	554.365	113	+0.229%
Re	587.330	106	-0.390%
Re #	622.254	100	-0.441%
Mi	659.255	95	+0.206%
Fa	698.457	89	-0.543%
Fa #	739.989	84	-0.548%
Sol	783.991	30	+0.350%
Sol #	830.609	75	-0.328%
La	880.000	71	-0.032%
La #	932.328	67	-0.055%
Si	987.767	63	-0.435%

Ottava 1

NOTA	FREQUENZA	PERIODO	ERRORE RELATIVO
Do	1046.502	60	+0.462%
Do #	1108.731	56	-0.662%
Re	1174.659	53	-0.390%
Re #	1244.508	50	-0.441%
Mi	1318.510	47	-0.855%
Fa	1396.913	45	+0.574%
Fa #	1479.978	42	-0.548%
Sol	1567.982	40	+0.350%
Sol #	1661.219	38	+0.992%
La	1760.000	36	+1.357%
La #	1864.655	34	+1.417%
Si	1975.533	32	+1.134%

Ottava 2

NOTA	FREQUENZA	PERIODO	ERRORE RELATIVO
Do	2093.004	30	+0.462%
Do #	2217.461	28	-0.662%
Re	2349.318	27	+1.469%
Re #	2489.016	25	-0.441%
Mi	2637.021	24	+1.246%
Fa	2793.826	22	-1.685%
Fa #	2959.955	21	-0.548%
Sol	3135.963	20	+0.350%
Sol #	3322.438	19	+0.992%
La	3520.000	18	+1.357%
La #	3729.310	17	+1.417%
Si	3951.066	16	+1.134%

Ottava 3

I valori sopra riportati sono tutti calcolati dal La internazionale nel modo seguente:

FREQUENZA = 440*(2^(OTTAVA+((n-10)/12)))

PERIODO = ROUND(62500/FREQUENZA)

... dove N è 1 per il Do, 2 per il Do #, 3 per il Re, ecc.

Parte 6: Messaggi di errore del BASIC

1 Unexpected NEXT

E' stato trovato un comando NEXT che non ha un corrispondente FOR, o la variabile di controllo nel comando NEXT non è uguale a quella del FOR.

2 Syntax Error

Il BASIC non comprende la linea in questione che contiene un costrutto non legale.

3 Unexpected RETURN

E' stato trovato un RETURN all'esterno di una sub-routine.

4 DATA exhausted

Un comando READ ha cercato di leggere oltre la fine dell'ultimo DATA

5 Improper argument

E' un errore di carattere generale. Il valore dell'argomento della funzione, o un parametro di un comando non è lecito per qualche motivo.

6 Overflow

Il risultato di una operazione aritmetica ha oltrepassato i limiti numerici della macchina. Può essere un overflow in virgola mobile, nel qual caso l'operazione ha fornito un valore più grande di 1.7 E^38 (circa). Altrimenti può essere il risultato di un tentativo fallito di convertire un numero in virgola mobile in un intero a 16 bit con il segno.

7 Memory full

Lo spazio occupato dal programma attualmente in uso o dalle sue variabili è semplicemente troppo grande o una struttura di controllo è innestata troppo profondamente (vale per i GOSUB, i WHILE ed i FOR).

Il comando MEMORY può dare questo errore se si cerca di dare alla memoria del BASIC uno spazio troppo piccolo o troppo grosso. Si noti che ad ogni file aperto corrisponde un buffer che può ridurre lo spazio di memoria utilizzabile dal comando MEMORY.

8 Line does not exist

Non si trova la linea a cui si fa riferimento.

9 Subscript out of range

Uno degli indici di un vettore è troppo grande o troppo piccolo.

10 Array already dimensioned

Uno dei vettori in una istruzione DIM è già stato dichiarato.

11 Division by zero

Può capitare nelle divisioni per numeri reali, interi, moduli interi o nelle esponenziali.

12 Invalid direct command

L'ultimo comando immesso non ha senso in modo diretto.

13 Type mismatch

E' stato presentato un valore numerico dove occorreva una stringa o viceversa, oppure è stato trovato da una istruzione READ o INPUT un numero composto irregolarmente.

14 String space full

Sono state create così tante stringhe che non vi è più spazio, anche dopo una "pulizia della memoria".

15 String too long

La stringa è più lunga di 255 caratteri. Può essere generato se si cerca di riunire insieme più stringhe.

16 String expression too complex

Le espressioni su stringhe possono generare un gran numero di stringhe intermedie. Quando il numero di tali stringhe eccede un limite ragionevole, viene riportato questo messaggio di errore.

17 Cannot CONTinue

Per qualche motivo il programma non può esser fatto ripartire usando CONT. CONT fa ripartire un programma dopo un comando STOP, dopo una sequenza [ESC][ESC] o dopo un errore e che ogni modifica apportata al programma rende il riavvio impossibile.

18 Unknown user function

Non vi è un DEF FN per la funzione chiamata.

19 RESUME missing

Si è incontrata la fine del programma in modo gestione errori (in una routine ON ERROR GOTO).

20 Unexpected RESUME

L'uso di RESUME è lecito solo in modo gestione errori (in una routine ON ERROR GOTO).

21 Direct command found

Caricando un file è stata trovata una linea senza numero di linea.

22 Operand missing

Il BASIC ha incontrato una espressione incompleta.

23 Line too long

Una linea, convertita nel formato interno del BASIC è diventata troppo lunga.

24 EOF met

E' stato fatto un tentativo di leggere oltre la fine di un file.

25 File type error

Il file che è stato letto non è di un tipo utilizzabile. OPENIN può leggere solo file ASCII. Analogamente LOAD, RUN, ecc. possono utilizzare solo file prodotti da un SAVE.

26 Next missing

Non si trova un NEXT che corrisponda ad un comando FOR. La linea che accompagna questo messaggio indica il FOR in cui si è rilevato l'errore.

27 File already open

E' stato eseguito un comando OPENIN o OPENOUT prima di chiudere un file già aperto.

28 Unknown command

Il BASIC non riesce a trovare un comando esterno, un comando preceduto dalla barra verticale.

29 WEND missing

Non si trova un WEND dopo un comando WHILE.

30 Unexpected WEND

Si è incontrato un WEND all'esterno di un ciclo WHILE o un WEND che non corrisponde ad un comando WHILE.

31 File not open

Vedere il seguente paragrafo intitolato "Errori relativi al disco".

32 Broken in

Vedere il seguente paragrafo intitolato "Errori relativi al disco".

Errori relativi al disco (AMSDOS)

Vi sono molti errori che possono capitare durante l'utilizzo dei dischi. Il BASIC può gestire tutti questi errori come ERRore numero 32, ma ulteriori informazioni vengono riportate dalla funzione DERR usata quando trova tale errore. Essa restituisce i seguenti valori:

Errore AMSDOS	Valore DERR	Causa
0	Ø o 22	E' stato premuto [ESC]
14	142 (128+14)	Il canale non è in uno stato utilizzabile
15	143 (128+15)	Raggiunta la fine hrdware del file
16	144 (128+16)	Comando errato, normalmete causato da un nome di file errato
17	145 (128+17)	Il file esiste già
18	146 (128+18)	Il file non esiste
19	147 (128+19)	La direttrice è piena
20	148 (128+20)	Il disco è pieno
21	149 (128+21)	Il disco è stato cambiato mentre erano ancora aperti dei file su di esso
22	15Ø (128+22)	Il file è a sola lettura
26	154 (128+26)	Raggiunta la fine software del file

Se AMSDOS ha già trovato un errore, il bit 7 è attivo; il valore di DERR viene perciò sommato a 128

Altri valori ritornati da DERR vengono prodotti dalla scheda di gestione del drive ed hanno sempre il bit 6 attivo. Il bit 7 indica che l'errore è stato rilevato dall'AMSDOS (come si è già spiegato). Il significato dei bit è il seguente:

Bit	**Significato**
0	Indirizzo mancante
1	Non scrivibile, il disco è protetto dalla scrittura
2	Nessun dato, non si trova il settore
3	Drive non pronto, nessun disco nel drive
4	Errore di overrun
5	Errore nei dati, errore di Ridondanza ciclica
6	Sempre a 1, indica che l'errore è prodotto dalla scheda di controllo dei dischi
7	1 se l'errore è già stato trovato da AMSDOS

ERR può anche restituire 31 se si è tentato l'accesso ad un file non aperto. Il normale modo in cui è possibile usare ERR e DERR consiste nell'includerlo in un comando ON ERROR GOTO che chiama una breve routine che controlla se ERR ha valore 31 o 32 e, se il valore è 32, è possibile interrogare DERR per ottenere ulteriori informazioni che riguardano la natura dell'errore. Ad esempio:

```
10 ON ERROR GOTO 1000
20 OPENOUT "miofile.asc"
30 WRITE #9, "dati di prova"
40 CLOSEOUT
50 END
1000 amsdoserr= (DERR AND &7F):REM elimina il bit 7
1010 IF ERR<31 THEN END
1020 IF ERR=31 THEN PRINT "sicuro di aver immesso correttamente la linea
20?":END
1030 IF amsdoserr=20 THEN PRINT "il disco è pieno, usare un nuovo
disco":END
1040 IF amsdoserr=&X01001000 THEN PRINT "inserire un disco nel drive, poi
premere un tasto":WHILE INKEY$="":WEND:RESUME
1050 END
```

Parte 7: Parole chiave del BASIC

Quello che segue è un elenco di tutte le parole riservate del BASIC del 464/6128. Le parole riservate non possono essere utilizzate per i nomi di variabili.

ABS, AFTER, AND, ASC, ATN, AUTO

BIN$, BORDER

CALL, CAT, CHAIN, CHR$, CINT, CLEAR, CLG, CLOSEIN, CLOSEOUT, CLS, CONT, COPYCHR$, COS, CREAL, CURSOR

DATA, DEC$, DEF, DEFINT, DEFREAL, DEFSTR, DEG, DELETE, DERR, DI, DIM, DRAW, DRAWR

EDIT, EI, ELSE, END, ENT, ENV, EOF, ERASE, ERL, ERR, ERROR, EVERY, EXP

FILL, FIX, FN, FOR, FRAME, FRE

GOSUB, GOTO, GRAPHICS

HEX$, HIMEM

IF, INK, INKEY, INKEY$, INP, INPUT, INSTR, INT

JOY

KEY

LEFT$, LEN, LET, LINE, LIST, LOAD, LOCATE, LOG, LOG10, LOWER$

MASK, MAX, MEMORY, MERGE, MID$, MIN, MOD, MODE, MOVE, MOVER

NEXT, NEW, NOT

ON, ON BREAK, ON ERROR GOTO 0, ON SQ, OPENIN, OPENOUT, OR, ORIGIN, OUT

PAPER, PEEK, PEN, PI, PLOT, PLOTR, POKE, POS, PRINT

RAD, RANDOMIZE, READ, RELEASE, REM, REMAIN, RENUM, RESTORE, RESUME, RETURN, RIGHT$, RND, ROUND, RUN

SAVE, SGN, SIN, SOUND, SPACE$, SPC, SPEED, SQ, SQR, STEP, STOP, STR$, STRING$, SWAP, SYMBOL

TAB, TAG, TAGOFF, TAN, TEST, TESTR, THEN, TIME, TO, TROFF, TRON

UNT, UPPER$, USING

VAL, VPOS

WAIT, WEND, WHILE, WIDTH, WINDOW, WRITE

XOR, XPOS

YPOS

ZONE

Parte 8: Tabelle

Tabella per il testo e la finestra - MODE 0 (20 colonne)

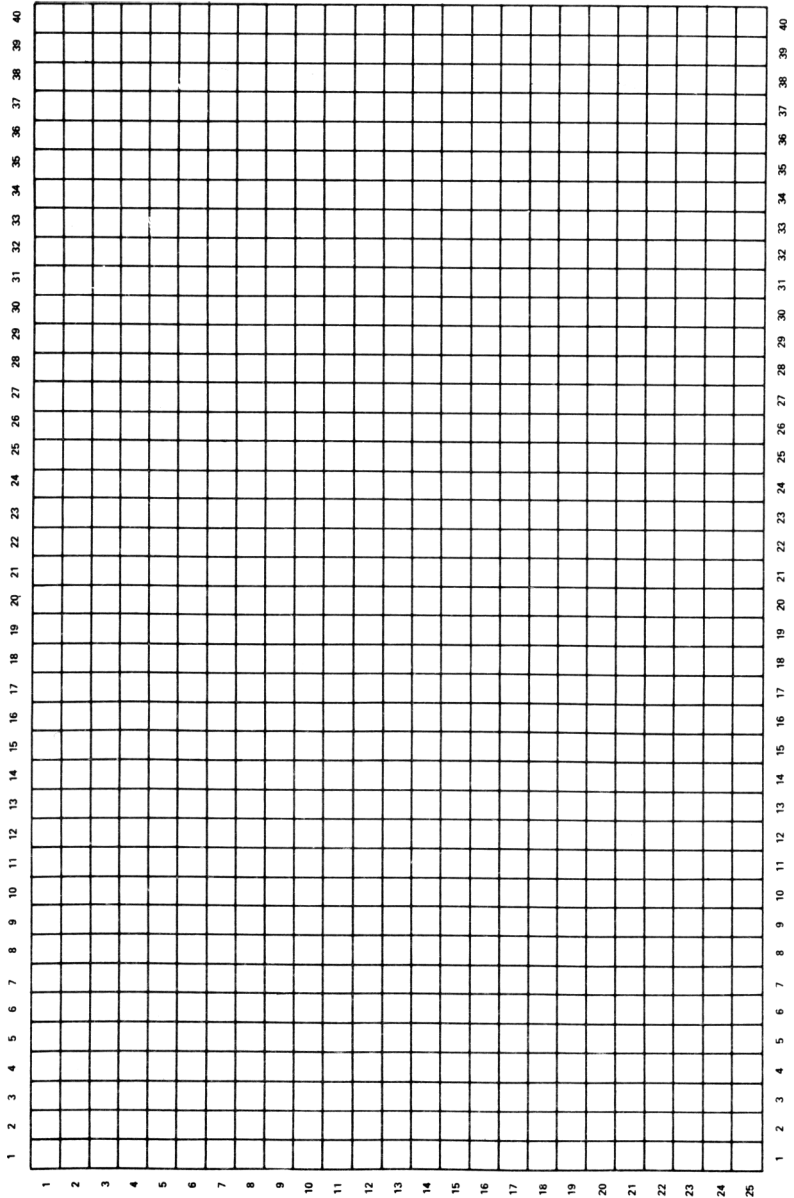
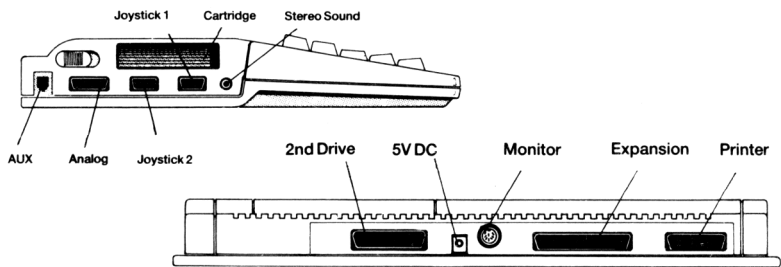
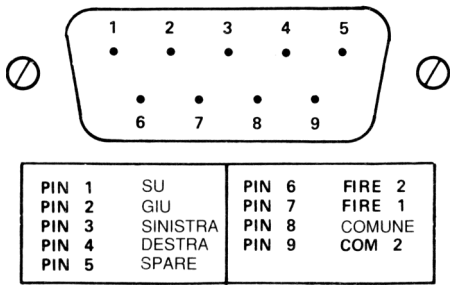



Tabella per il testo e la finestra - MODE 2 (80 colonne)

Tabella per gli inviluppi e la musica

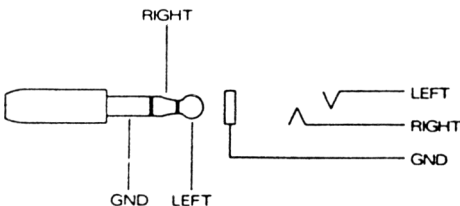
Parte 9: Collegamenti

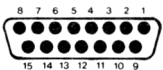
Prese di Input e Output del 464/6128

Presa AUX

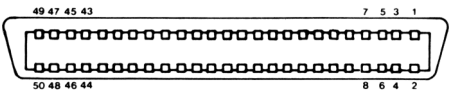
Presa per Joystick

NOTA: SUL CONNETTORE JOYSTICK 2 IL PIN 9 NON E' COM2

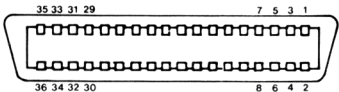
Presa per monitor

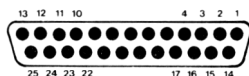
PIN 1	$\overline{\text{SYNC}}$	PIN 5	BLUE
PIN 2	GREEN	PIN 6	L SOUND
PIN 3	LUM	PIN 7	R SOUND
PIN 4	RED	PIN 8	GND

Uscita stereo

PIN 1 LEFT CHANNEL
PIN 2 RIGHT CHANNEL
PIN 3 GND

Presa analogica

JOYSTICK ANALOGICO 1		JOYSTICK ANALOGICO 2	
PIN 1	GND (POT COMMON)	PIN 9	GND (POT COMMON)
PIN 2	FIRE 1	PIN 10	FIRE 1
PIN 3	X1	PIN 11	X2
PIN 4	COM1 (SWITCHES)	PIN 12	COM2 (SWITCHES)
PIN 5	+5V	PIN 13	Y2
PIN 6	Y1	PIN 14	FIRE 2
PIN 7	FIRE 2	PIN 15	GND (POT COMMON)
PIN 8	GND (POT COMMON)		

Pettine di espansione

PIN 1	SUONO	PIN 18	A0	PIN 35	INT
PIN 2	TERRA	PIN 19	D7	PIN 36	NMI
PIN 3	A15	PIN 20	D6	PIN 37	BUSR2
PIN 4	A14	PIN 21	D5	PIN 38	BUSAK
PIN 5	A13	PIN 22	D4	PIN 39	READY
PIN 6	A12	PIN 23	D3	PIN 40	BUS RESET
PIN 7	A11	PIN 24	D2	PIN 41	RESET
PIN 8	A10	PIN 25	D1	PIN 42	ROMEN
PIN 9	A9	PIN 26	D0	PIN 43	ROMDIS
PIN 10	A8	PIN 27	+ 5v	PIN 44	RAMRD
PIN 11	A7	PIN 28	MREQ	PIN 45	RAMDIS
PIN 12	A6	PIN 29	M1	PIN 46	CURSOR
PIN 13	A5	PIN 30	RFSH	PIN 47	L. PEN
PIN 14	A4	PIN 31	IORQ	PIN 48	EXP
PIN 15	A3	PIN 32	RD	PIN 49	GND
PIN 16	A2	PIN 33	WR	PIN 50	φ
PIN 17	A1	PIN 34	HALT		

Presa per secondo drive (solo 6128)

PIN 1		PIN 2	GND
PIN 3		PIN 4	GND
PIN 5		PIN 6	GND
PIN 7	INDEX	PIN 8	GND
PIN 9		PIN 10	GND
PIN 11	DRIVE 1 SELECT	PIN 12	GND
PIN 13		PIN 14	GND
PIN 15	MOTOR ON	PIN 16	GND
PIN 17	DIRECTION SELECT	PIN 18	GND
PIN 19	STEP	PIN 20	GND
PIN 21	WRITE DATA	PIN 22	GND
PIN 23	WRITE GATE	PIN 24	GND
PIN 25	TRACK Ø	PIN 26	GND
PIN 27	WRITE PROTECT	PIN 28	GND
PIN 29	READ DATA	PIN 30	GND
PIN 31	SIDE 1 SELECT	PIN 32	GND
PIN 33	READY	PIN 34	GND
PIN 35		PIN 36	GND

Porta stampante

PIN 1	$\overline{\text{STROBE}}$	PIN 17	GND
PIN 2	D0	PIN 18	GND
PIN 3	D1	PIN 19	GND
PIN 4	D2	PIN 20	GND
PIN 5	D3	PIN 21	GND
PIN 6	D4	PIN 22	GND
PIN 7	D5	PIN 23	GND
PIN 8	D6	PIN 24	GND
PIN 9	D7	PIN 25	GND
PIN 11	BUSY		
PIN 16	+5V	Gli altri pin non sono connessi	

Parte 10: Stampanti

Interfacciamento stampanti

Il 464/6128 permette di collegare ed usare tutte le stampanti che usano l'interfaccia standard Centronics.

Il cavo per la stampante è costituito semplicemente da una connessione uno-a-uno tra la porta stampante sul retro del computer ed il connettore di una stampante parallela.

I dettagli relativi alle connessioni presenti sull'interfaccia si trovano nel Capitolo 9.

Il computer usa il segnale BUSY (pin 11) per sincronizzarsi con la stampante; in tal modo si metterà in attesa se la stampante è fuori-linea (OFF-LINE).

Non vi sono comandi di predisposizione a carico dell'utente, e l'output verrà sempre inviato al canale #8.

Sebbene la porta per stampante del 464/6128 sia abilitata all'uso delle stampanti a matrice di basso costo, con una interfaccia adatta, potrà utilizzare stampanti a margherita, plotter grafici e stampanti a getto di inchiostro a più colori. La compatibilità è garantita dall'interfaccia parallela standard.

Configurazione della stampante

Viene fornita la possibilità di stampare alcuni caratteri che possono apparire sullo schermo e che sono disponibili sulle stampanti ma con codici diversi. La maggior parte di questi simboli sarà disponibile solo se la stampante è predisposta ad utilizzare i caratteri di una data lingua. Ad esempio:

```
PRINT CHR$(&A0)
^
PRINT #8, chr$(&A0)
^ viene stampato sulla stampante.
```

Ciò funzionerà anche se il codice per l'accento circonflesso sulla stampante è &5E. In altre parole la routine di stampa ha riconosciuto &A0 come uno dei codici contenuti nella tabella di traduzione e l'ha convertito in &5E in modo che lo stesso carattere che appare sullo schermo venga anche riportato sulla stampante. Il codice &5E produrrà sulla stampante un accento circonflesso, per qualunque linguaggio essa sia predisposta (ciò non è vero per tutti i caratteri della tabella di traduzione). Gli altri caratteri sottoposti a traduzione sono riportati nella seguente tabella.

CHR$	Carattere sullo schermo	Traduzione sulla stampante	GB	U.S.A.	Francia	Germania	Spagna
&A0	^	&5E	^	^	^	^	^
&A2	¨	&7B	†	†	†	†	¨
&A3	£	&23	£	#	#	#	Pt
&A6	§	&40	†	†	†	§	†
&AE	¿	&5D	†	†	†	†	¿
&AF	¡	&5B	†	†	†	†	¡

† Per i caratteri stampati, si faccia riferimento al manuale della stampante.

Questa tabella riporta una parte delle traduzioni e può essere modificata se è il caso.

Parte 11: Joystick e Paddle

Il software del computer può gestire uno o due joystick. Essi vengono gestiti come una parte della tastiera ed in tal modo possono essere interrogati mediante INKEY e INKEY$.

Si noti che nella maggior parte dei casi, il tasto principale "fire" di un joystick viene interpretato come "fire 2"

Le funzioni Joy(0) e Joy (1) permettono il controllo rispettivamente del primo e del secondo joystick. La funzione restituisce un risultato mappato bit a bit che indica lo stato degli interruttori del joystick all'ultima scansione della tastiera.

La tabella qui sotto indica i valori restituiti da entrambi i joystick. I valori della funzione JOY sono seguiti dai valori da utilizzare nei comandi che usano come parametro il numero del tasto (ad esempio INKEY e KEY DEF).

STATO	COMANDO JOY		VALORE DEL TASTO		
	BIT USATO	VALORE RESTITUITO	PRIMO JOYSTICK	SECONDO JOYSTICK	TASTO EQUIVALENTE
Alto	0	1	72	48	'6'
Basso	1	2	73	49	'5'
Sinistra	2	4	74	50	'R'
Destra	3	8	75	51	'T'
Fire 2	4	16	76	52	'G'
Fire 1	5	32	77	53	'F'

Si noti che quando i valori dei tasti del secondo joystick vengono restituiti, il computer non può sapere se tali valori sono stati provocati dal joystick o da uno degli equivalenti tasti della tastiera (indicati nell'ultima colonna della tabella precedente). Ciò significa che la tastiera può essere usata per sostituire il secondo joystick.

Parte 12: Organizzazione dei dischi

Il BIOS permette di utilizzare tre diversi formati di dischi: il formato SYSTEM, il formato DATA ONLY ed il formato IBM. Sotto AMSDOS, il formato di un disco è controllato automaticamente ogni volta che si fa accesso ad un disco che non ha file aperti. Per permettere l'uso del controllo automatico, ogni formato ha un diverso tipo di numerazione dei settori.

I dischi da 3 pollici vengono utilizzati sulle due facce, ma una faccia per volta può essere utilizzata, a seconda del lato in cui viene inserito il disco. Vi può essere un diverso formato per lato.

Caratteristiche comuni a tutti i formati:

Singola faccia.
Dimensione dei settori: 512 byte.
40 tracce numerate tra 0 e 39.
Dimensione del blocco CP/M di 1024 byte.
Posto per 64 file nella direttrice.

Formato SYSTEM

9 settori per traccia numerati da &41 a &49
2 tracce riservate

Il formato SYSTEM è il principale formato supportato, in quanto il sistema operativo CP/M può essere caricato solo da un disco in formato SYSTEM.

	CP/M Plus
Traccia 0 settore &41	:settore di boot
Traccia 0 settore &42	:non usato
Traccia 0 settori da &43 a &47	:non usato
Traccia 0 settori &48 e &49	:non usato
Traccia 1 settori da &41 a &49	:non usato

Il formato VENDOR è una versione speciale del formato SYSTEM che non contiene software di sistema nelle prime 2 tracce riservate. Viene utilizzato nel campo della distribuzione di software.

Formato DATA ONLY

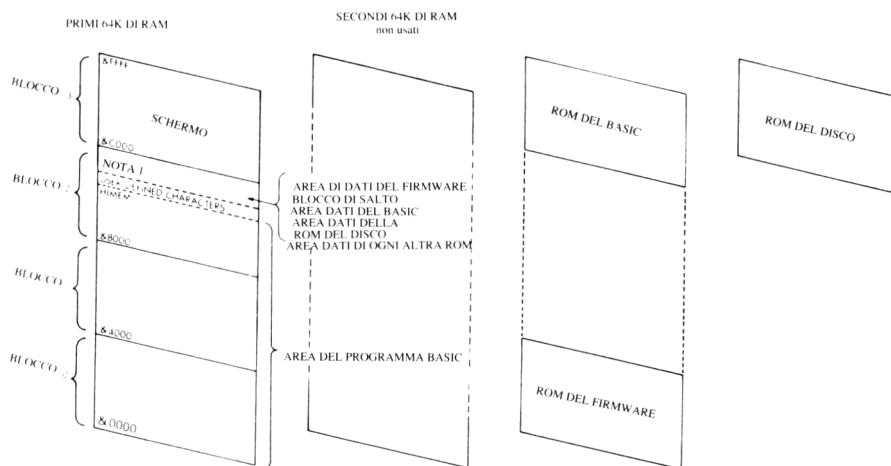
9 settori per traccia numerati da &C1 a &C9
0 tracce riservate

Parte 13: Sistema di estensione residente RSX

I comandi esterni sono stati introdotti nel Capitolo 5 (che riguarda il sistema operativo AMSDOS). Principalmente, un comando esterno costituisce un modo per estendere il dizionario del BASIC aggiungendo nuovi comandi che devono essere preceduti dal prefisso "|". Le istruzioni per la macchina per i nuovi comandi di AMSDOS sono contenuti in una ROM, e la necessaria gestione dell'aggiunta di nuovi comandi viene effettuata automaticamente quando il 464/6128 avvia il BASIC.

E' anche possibile aggiungere altri comandi esterni (dopo che il BASIC è stato avviato) caricando le istruzioni macchina in RAM. Tali comandi vengono chiamati RSX e funzionano esattamente nello stesso modo dei comandi residenti su ROM. I comandi RSX devono essere caricati da disco (o da cassetta) ogni volta che il 464/6128 avvia o reinizializza il BASIC. Normalmente i comandi RSX verranno utilizzati per gestire speciali dispositivi periferici come una penna ottica o un sintetizzatore vocale.

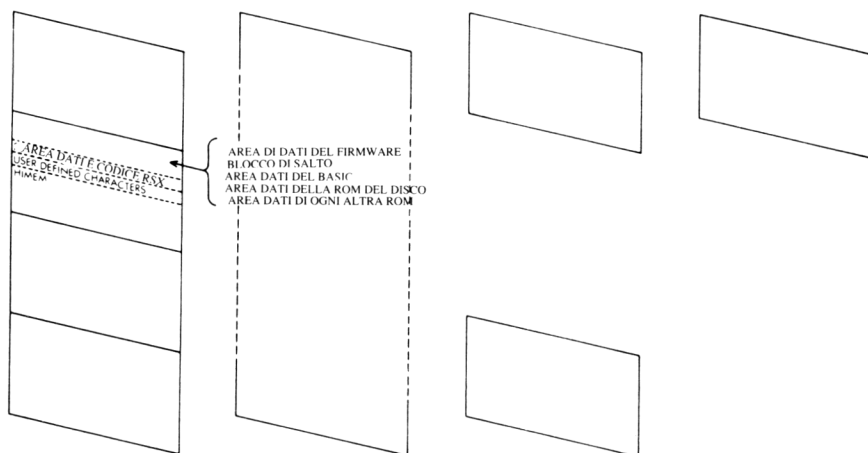
Parte 14: Memoria

Il 6128 contiene 128K di RAM e 48K di ROM. Questa memoria è disponibile al BASIC 1.1 nel modo seguente. I primi 64K di RAM sono divisi in 4 blocchi (ognuno di 16K) numerati da 0 a 3. Lo schermo usa il Blocco 3 e la parte superiore del Blocco 2 è utilizzata dalle variabili di sistema.

Mappa di memoria del BASIC 1.1

I caratteri definibili dall'utente vengono inizialmente posti sopra HIMEM. HIMEM può essere modificato con il comando MEMORY ma viene abbassato automaticamente di 4K per creare un buffer per i file di AMSDOS. Il numero dei caratteri definibili dall'utente può essere modificato solo se HIMEM non è cambiato dall'ultima volta che essi sono stati fissati (a meno che l'ultima volta che sono stati fissati era stato scelto di non avere alcun carattere definibile dall'utente con il comando SYMBOL AFTER 256). Quando il BASIC viene avviato, i caratteri definibili dall'utente sono fissati come se fosse stato usato il comando SYMBOL AFTER 240.

E' perciò prudente ridurre l'area dei caratteri definibili dall'utente prima di cambiare permanentemente HIMEM e poi ristabilire i caratteri definibili dall'utente nella nuova posizione. In questo modo i programmi successivi potranno alterare l'assegnamento SYMBOL AFTER.

L'esempio qui sotto mostra questo schema all'opera quando HIMEM è abbassato in congiunzione al caricamento di un RSX.

```
100 SYMBOL AFTER 256 ' collapse user defined characters
110 rsxaddress=HIMEM-rsxlength
120 MEMORY rsxaddress-1
130 LOAD "rsxcode",rsxaddress
140 CALL rsxaddress ' log on rsx
150 SYMBOL AFTER 140 ' restore user defined characters
```

Mappa di memoria con gli RSX caricati nella giusta posizione

I/O Aggiuntivo

La maggior parte degli indirizzi della porta di I/O è riservata dal computer; in particolare gli indirizzi sotto &7FFF non devono essere usati.

Si intende che la parte degli indirizzi da A0 ad A7 dovrebbe riflettere il tipo di dispositivo di I/O e che le linee di indirizzi A8 e A9 possono essere decodificate per selezionare i registri all'interno del dispositivo di I/O. Delle rimanenti linee di indirizzi, solo A10 deve essere decodificato (come basso) mentre le linee da A11 ad A15 sono alte. Così ogni dispositivo può avere i registri indirizzati come &F8??, &F9??, &FA?? e &FB?? dove ??è compreso tra DC e DF per le interfacce di comunicazione e va da E0 a FE per dispositivi di altro genere.

Si noti che devono essere usate le istruzioni Z80 che portano il registro B nella parte superiore del bus degli indirizzi (A15-A8).

ROM Aggiuntive

Si è previsto l'uso di ROM aggiuntive al posto di una qualunque delle ROM interne. La decisione degli indirizzi e la logica di selezione dei banchi sarà contenuta in un modulo connesso al bus di espansione, ma tutti i segnali richiesti vengono portati al bus di espansione.

Parte 15: Emulatore di terminale di CP/M Plus

Nella parte 1 di questo capitolo è stata illustrata una tabella di tutti i caratteri di controllo (insieme alle rispettive operazioni). Queste azioni vengono eseguite se i caratteri vengono stampati sullo schermo da BASIC o da CP/M 2.2 e sono state scelte per essere di uso semplice e per utilizzare le possibilità di gestione dello schermo del firmware. Queste caratteristiche sono specifiche dei computer AMSTRAD ed il software dovrà quindi essere adattato al loro uso.

Nell'ambiente degli affari e commerciale del software di CP/M Plus, è normale aspettarsi l'esistenza di gestioni standard dello schermo in modo che il software sia facilmente trasportabile ed installabile su diverse macchine. L'implementazione di CP/M Plus sul 6128 include un emulatore di terminale che fornisce caratteristiche molto simili a quelle dello Zenith Z19/Z29. La procedura di installazione del software del CP/M Plus include normalmente, come standard, questo tipo di terminale.

Le caratteristiche offerte dall'emulatore di terminale di CP/M Plus includono molte di quelle precedentemente offerte dal firmware di gestione dello schermo anche se occorrono diversi codici di controllo.

Vi è anche un numero considerevole di nuove e più sofisticate possibilità.

I caratteri tra &20 e &FF vengono visualizzati all'attuale posizione del cursore. Se il cursore non è all'estrema destra dello schermo, esso verrà portato di una colonna più a destra. Se il cursore è all'estrema destra dello schermo ed è permesso l'"A capo", il cursore verrà portato nella prima colonna della linea successiva, facendo, se necessario, scorrere lo schermo.

I caratteri che vanno da &00 a &1F vengono interpretati come codici di controllo nel modo seguente.

&07	BEL	Emette un bip
&08	BS	Indietro. Sposta a sinistra di una colonna. Se il cursore è nella prima colonna a sinistra ma non in cima allo schermo e l'"A capo" è abilitato, esso verrà portato sulla colonna più a destra della riga precedente.
&0A	LF	Linefeed. Sposta il cursore verso il basso di una linea, facendo scorrere lo schermo se necessario.
&0D	CR	Ritorno carrello. Porta il cursore nella prima colonna della riga in cui si trova.
&1B	ESC	Inizia una sequenza di Escape.

Tutti gli altri codici di controllo vengono ignorati.

Vengono riconosciute le seguenti sequenze di Escape. Ogni altro carattere che segue l'ESC verrà visualizzato ed il cursore verrà fatto avanzare. Questa caratteristica può essere sfruttata per visualizzare i caratteri tra &00 e &1F. Si noti che molti linguaggi applicativi espandono il codice di controllo &09 (TAB) in un certo numero di spazi e quindi la sequenza [ESC][TAB] spesso potrà non visualizzare il carattere &09.

[ESC]0 — Disabilita la linea di stato. I messaggi relativi al disco appariranno sullo schermo che potrà usare l'ultima linea dello schermo.

[ESC]1 — Abilita la linea di stato. I messaggi relativi al disco appariranno sull'ultima linea dello schermo.

[ESC]2n — Cambia il set di caratteri (vedere parte 16 di questo capitolo). "n" è il parametro relativo al linguaggio mascherato con &07. Alcune matrici di caratteri tra &20 e &7F vengono sostituiti da altri caratteri con codici che vanno da &80 a &FF. L'azione di questo comando è molto simile a quella usata per selezionare un diverso set di caratteri sulle stampanti che prevedono tale possibilità.

n=0 USA
n=1 Francia
n=2 Germania
n=3 Gran Bretagna
n=4 Danimarca
n=5 Svezia
n=6 Italia
n=7 Spagna

[ESC]3"m"	Cambia modalità di schermo. "m"= modo schermo + &20. Il valore viene mascherato con &03 per fornire i modi da 0 a 2. Il modo 3 viene ignorato. Lo schermo viene cancellato ma il cursore non viene spostato.
[ESC]A	Cursore in alto. Nella riga superiore non fa nulla.
[ESC]B	Cursore in basso. Nella riga inferiore non fa nulla.
[ESC]C	Cursore avanti. Nella colonna più a destra non fa nulla.
[ESC]D	Cursore indietro. Nella colonna più a sinistra non fa nulla.
[ESC]E	Cancella la pagina. La posizione del cursore non viene modificata. Questo comando cancella l'intero schermo, anche in modo 24x80. Altre sequenze di escape cancellano la sola area 24x80 se il computer si trova in tale modo.
[ESC]H	Cursore nell'angolo in alto a sinistra.
[ESC]I	Sposta il cursore in alto di una riga, facendo scorrere la finestra se necessario.
[ESC]J	Cancella la fine della pagina, partendo dal carattere che si trova alla posizione del cursore (compreso). La posizione del cursore non viene modificata.
[ESC]K	Cancella la fine della riga, partendo dal carattere che si trova alla posizione del cursore (compreso). La posizione del cursore non viene modificata.
[ESC]L	Inserisce una linea. Tutte le linea da quella del cursore (compresa) in poi vengono fatte scorrere verso il basso. La linea del cursore viene cancellata. La posizione del cursore non viene modificata.
[ESC]M	Cancella una linea. Tutte le linee, compresa quella del cursore, vengono fatte scorrere verso l'alto. L'ultima riga viene cancellata. La posizione del cursore non viene modificata.
[ESC]N	Cancella un carattere. Tutti i caratteri a destra del cursore vengono spostati di una posizione a sinistra. Il carattere a sinistra della riga viene cancellato. La posizione del cursore non viene modificata.
[ESC]Y "r" "c"	Sposta il cursore in una data posizione. Se la posizione è oltre i limiti dello schermo, sposta sul margine il cursore. r=riga+&20, c=colonna+&20. il primo carattere in alto a sinistra è nella riga 0 e colonna 0.

[ESC]b "pc" — Fissa il colore dei caratteri. Riguarda tutti i caratteri contenuti nello schermo. pc è il parametro colore mascherato con &3F e utilizzato come tre numeri a 2-bit che specificano l'intensità dei tre colori fondamentali: i bit 0 e 1 per il blu, i bit 2 e 3 per il rosso ed i bit 4 e 5 per il verde. Il 6128 permette tre livelli di intensità secondo lo schema:

464/6128	Intensità zero	Intensità media	Piena intensità
Bit colore	00 binario	01 o 10 binario	11 binario

[ESC]c "pc" — Fissa il colore di fondo. Riguarda tutto lo schermo e la cornice dello schermo. Il colore viene specificato come sopra.

[ESC]d — Cancella fino all'inizio della pagina compreso il carattere nella posizione del cursore. La posizione del cursore non viene modificata.

[ESC]e — Abilita il simbolo del cursore. Per evitare lampeggi indesiderati, il cursore non viene acceso durante l'output del testo ma 1/10 di secondo dopo la scrittura dell'ultimo carattere.

[ESC]f — Disabilita il simbolo del cursore.

[ESC]j — Salva la posizione del cursore.

[ESC]k — Riporta il cursore nella posizione precedentemente salvata con [ESC]j

[ESC]l — Cancella la linea. La posizione del cursore non viene modificata.

[ESC]o — Cancella l'inizio della linea compreso il carattere alla posizione del cursore. La posizione del cursore non viene modificata.

[ESC]p — Entra in modo video inverso. I caratteri vengono stampati con i colori del fondo e viceversa.

[ESC]q — Esce dal modo video inverso.

[ESC]r — Entra in modo sottolineato (non possibile sul 6128).

[ESC]u — Esce dal modo sottolineato (non possibile sul 6128).

[ESC]v — A capo alla fine della linea.

[ESC]w Getta via i caratteri se il cursore è alla fine della linea.

[ESC]x Entra in modo 24x80. Alcuni programmi applicativi possono richiedere uno schermo standard 24x80. Questo comando abilita tale schermo senza preoccuparsi dell'intera dimensione dello schermo che può variare da macchina a macchina da paese a paese e dalla presenza della linea di stato. Lo schermo viene cancellato.

[ESC]y Esce dal modo 24x80. Lo schermo viene cancellato.

Parte 16: Set di caratteri di CP/M Plus

Nella Parte 10 di questo capitolo è stata descritta una tabella di traduzione per la stampante. Lo scopo di tale tabella è quello di convertire alcuni caratteri del BASIC in una forma in cui possano essere stampati con la stampante predisposta per una determinata lingua. Questa possibilità è però limitata dal fatto che pochi caratteri nazionali appaiono nel set di caratteri del BASIC.

Sebbene questo schema di traduzione per la stampante possa essere utilizzato anche in CP/M Plus, il set di caratteri è stato potenziato per permettere una corrispondenza quasi completa tra i caratteri dello schermo e quelli della stampante. L'unico simbolo non permesso sullo schermo è il simbolo della moneta svedese, sostituito dal $. La tabella qui sotto conferma questo fatto:

Set di caratteri internazionali di CP/M Plus

L'utilizzo della macchina in un ambiente straniero richiede due azioni:

1. La stampante deve essere predisposta per la lingua considerata, normalmente utilizzando dei micro-interruttori ma alcune stampanti utilizzano codici di controllo.

2. Deve essere richiamato il set di caratteri per lo schermo o mediante il comando:

   LANGUAGE n

   o inviando

   [ESC]2 n all' emulatore di terminale.

In pratica l'inizializzazione può essere effettuata all'interno del file PROFILE.SUB usando LANGUAGE e SETLST. CP/M Plus viene fornito con un ambiente di utilizzo americano, principalmente perchè la tastiera riporta un # sopra il tasto 3. Nell'elaborazione di testi si potrà utilizzare l'ambiente italiano.

Software a 7 bit ...

Anche se la possibilità di usare i caratteri di varie lingue è molto potente, è importante far notare che i caratteri americani sostituiti dai caratteri locali del paese non saranno più utilizzabili. Questo è un compromesso a cui si deve giungere se si usa software a 7 bit. Quasi tutto il software disponibile (comprese molte utility di CP/M Plus, programmi di elaborazione del testo e linguaggi) usa set di caratteri a solo 7 bit. Non sempre tutte le trasformazioni saranno di facile uso ed ognuno dovrà effettuare mentalmente le sostituzioni dei caratteri.

Sfortunatamente, le trasformazioni di alcune lingue sostituiscono caratteri come la barra verticale e le parentesi quadre e sebbene la possibilità di utilizzare le accentate aumenti la leggibilità del testo, in situazioni in cui occorre utilizzare le barre e le parentesi quadre come in DIR **[FULL]**, la leggibilità potrebbe diminuire moltissimo. Si ricordi che il programma applicativo utilizza i codici ASCII, indipendentemente dai caratteri visualizzati sullo schermo. Il problema sta nel fatto che con 7 bit non vi sono codici sufficienti per tutti i caratteri.

Utilizzo di set di caratteri ad 8 bit

Il set di caratteri del BASIC ha 256 simboli diversi; (i valori da 128 a 255 (da &80 a &FF) contengono vari simboli grafici utilizzati in giochi e nei vari programmi personali (omini danzanti, cuori/quadri/picche/fiori, ecc.). Anche il CP/M Plus permette di utilizzare 256 caratteri ma i secondi 128 sono diversi da quelli del BASIC e riflettono l'utilizzo in campo economico e internazionale di CP/M Plus.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	∞	⊙	Γ	Δ	⊗	×	÷	∴	╥	↓	Σ	←	→	±	↔	Ω
1	α	β	ϑ	δ	ε	θ	λ	μ	π	ρ	σ	τ	φ	χ	ψ	ω
2		!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	◊

Caratteri da 0 a 127 (da &00 a &7F)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	▪	╧	╟	╚	╤	║	╔	╠	╢	╝	═	╩	╗	╣	╦	╬
9	·	╵	╴	└	╷	│	┌	├	╶	┘	─	┴	┐	┤	┬	┼
A	ª	º	°	£	©	¶	§	†	¼	½	¾	«	»	Pt	¿	¡
B	ƒ	¢	¨	´	^	‰	⅛	⅜	⅝	⅞	ß	○	●	¥	®	™
C	Á	É	Í	Ó	Ú	Â	Ê	Î	Ô	Û	À	È	Ì	Ò	Ù	Ÿ
D	Ä	Ë	Ï	Ö	Ü	Ç	Æ	Å	Ø	Ñ	Ã	Õ	≥	≤	≠	≃
E	á	é	í	ó	ú	â	ê	î	ô	û	à	è	ì	ò	ù	ÿ
F	ä	ë	ï	ö	ü	ç	æ	å	ø	ñ	ã	õ	⇒	⇐	⇔	≡

Caratteri da 128 a 255 (da &80 a &FF)

Set di caratteri standard (USA) di CP/M Plus

Il software che permette di utilizzare i caratteri ad 8 bit può usare questa tabella per avere a disposizione tutti i caratteri internazionali senza dover cambiare necessariamente lingua. In questo modo saranno disponibili insieme agli altri anche la barra verticalke e le parentesi quadre.

E' da notare tuttavia che pochi programmi attualmente usano i codici ad 8 bit e che i caratteri da 0 a 31 e da 128 a 255 appaiono sempre nella forma qui indicata. Le stampanti potranno avere un set di caratteri diverso e modi diversi di stampare lo stesso carattere.

Capitolo 7 - Qualcosa di più su BANK MANAGER (solo 6128)

Estensioni al BASIC che permettono di accedere al secondo lotto di 64K di RAM.

Argomenti trattati:

* Memorizzazione di immagini
* Operazioni su pseudo-file

La mappa di memoria di BASIC 1.1 (descritta nel Capitolo 6 parte 14) indica che 64K dei 128K di RAM è inutilizzata. Il BASIC e il firmware stesso risiedono in ROM, la quale insieme al disco ROM, aumenta la memoria disponibile da 64K a 112K (64K RAM, 48K ROM).

Ogni sezione di 16K è detta "blocco" e una qualsiasi sezione di quattro blocchi (che in totale fa 64K) è detta "banco". La tecnica di scelta dei blocchi è del resto chiamata "selezione dei banchi".

Il microprocessore Z80 può usare solo 64K di memoria per volta, quindi il sistema operativo contiene istruzioni per utilizzare il firmware ROM invece del Blocco 0 di RAM, e di utilizzare sia il BASIC ROM o il disco ROM invece del Blocco 3. Questa commutazione viene fatta automaticamente quando il BASIC o il firmware è richiesto. La scelta dei banchi di RAM estende questo concetto per includere la scelta della RAM piuttosto che la ROM. La commutazione viene effettuata da un programma assembler.

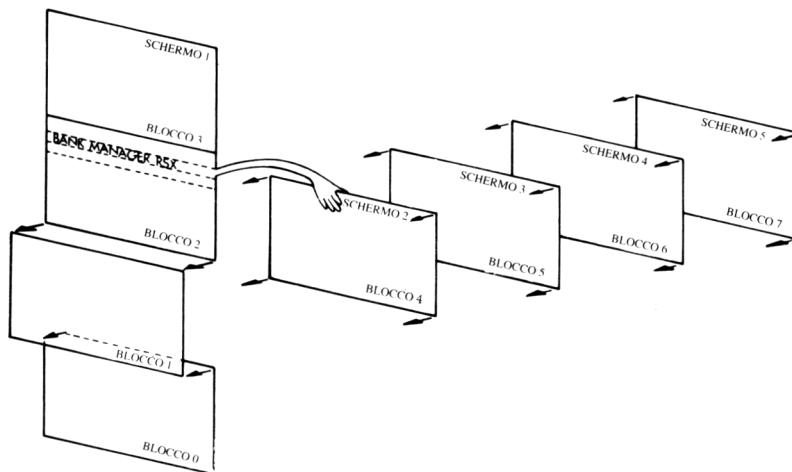
Sul lato 1 (Side 1) del disco di sistema viene fornito il programma BANKMAN.BAS. Se il programma viene eseguito dopo BASIC questo installerà il gestore di banchi RSX. Tale programma è noto come BANK MANAGER.

Un uso della memoria del secondo lotto di 64K consiste nella memorizzazione temporanea degli schermi grafici. Una tale applicazione può includere, ad esempio, un programma di disegno che memorizza un numero di schermi diversi o un video gioco che ha diverse schermate grafiche.

Un altro uso di tale memoria consiste nell'ampliare lo spazio di lavoro disponibile, il quale può essere visto come una estensione dello spazio di un vettore stringa o un semplice "disco RAM".

Parte 1: Memorizzazione di schermi grafici

Si scelga lo schermo...

BANK MANAGER è in grado di commutare la sua attenzione dal Blocco 1 ad uno dei quattro blocchi del secondo lotto di 64K. Il diagramma sotto illustra proprio questo concetto. Si noti come ogni blocco del secondo lotto di 64K occupa lo stesso spazio indirizzabile (&4000 fino a &7FFF). Il contenuto del Blocco 1 (probabilmente la metà del proprio programma!) è riservato, ed è memorizzato quando BANK MANAGER ha terminato la propria esecuzione. Vi sono altri tre banchi di selezione possibili (oltre ai cinque mostrati qui sotto) ma essi sono utili solo alla implementazione di CP/M Plus.

Selezione dei banchi hardware

Il BANK MANAGER ha inoltre due comandi per spostare gli schermi di informazione tra un blocco e un altro. I Blocchi dal 4 al 7 vengono commutati automaticamente quando richiesto e la memoria mappata è lasciata insieme al Blocco 1 attivo.

Il comando:

|SCREENSWAP,[<sezione schermo>,]<numero schermo>,<numero schermo>

...scambia i contenuti di due blocchi, mentre:

|SCREENCOPY,[<sezione schermo>,]<numero schermo destinazione>,<numero schermo origine>

...copia il contenuto di un blocco in un altro blocco.

Il parametro opzionale <sezione schermo> fa sì che il software copi solo 1/64 di blocco (256 byte di 16K). Tale parametro, del resto, assume valori compresi tra 0 e 63. Questo modo operativo è utile se viene richiesto di interfogliare un qualunque altro processo con il movimento dello schermo. La modifica di schermo può avvenire in 150/300-esimi di secondo.

Il <numero schermo> richiesto è 1 (schermo normale) e poi sia 2,3,4, o 5. Le operazioni di copia e di modifica quando lo schermo origine e quello destinazione è lo schermo 1 saranno molto più veloci. Si faccia molta attenzione agli effetti hardware di rotazione dello schermo. Esso deve essere predisposto in modo che tutti gli schermi grafici siano visti con lo schermo 1 disposto nella stessa posizione hardware. La posizione più semplice (per difetto) è quella fissata da un comando MODE.

Comandi di selezione dello schermo....

Si esegua, prima di tutto il programma BANK MANAGER presente sul lato 1 del disco di sistema, scrivendo:

RUN "BANKMAN"

Poi si scriva:

MODE 1

Lo schermo viene pulito. Si scriva ora:

'Questo è lo schermo originale
|SCREENCOPY,3,1 'Invia lo schermo originale alla memoria 3

CLS

Lo schermo viene nuovamente pulito. Si scriva ora:

```
'Questo è lo schermo intermedio
|SCREENCOPY,2,1 'Invia lo schermo intermedio alla memoria 2
|SCREENSWAP,2,3 'Scambia la memoria 2 e la 3
|SCREENCOPY,1,3 'Ripristina lo schermo intermedio della memoria 3
|SCREENCOPY,1,2 'Ripristina lo schermo originale dalla memoria 2
```

Infine, l'ultimo parte del Capitolo 9 include un programma 'Screen Design' che incorpora le caratteristiche di scelta dello schermo di BANK MANAGER.

Parte 2: Operazioni su pseudo-file

Ancora i file...

Quando si considera tutto il secondo lotto di 64K come un disco RAM, esso viene diviso in 'fileRAM' che comprende diversi recordi di lunghezza fissa. La lunghezza del record può assumere valori compresi tra 0 e 255 bytes, sebbene 2 byte sia la misura minima consigliata. Una volta stabilita la lunghezza del record RAM, si può accedere ad ogni record tramite il <numero RAMrecord>. E' perfettamente accettabile la scrittura del file RAM usando una lunghezza di record e leggerla usando un'altra.

NOTA: Il file RAM deve contenere solo dati; NON esiste una caratteristica che permetta di contenere istruzioni di programma.

Come è comune con l'accesso al diso random, vi è qui il concetto di 'numero di record attuale'. Ciò fornisce, in effetti, un numero di record per difetto, il quale è particolarmente utile quando vi è un passaggio automatico nel file RAM.

Il comando:

|BANKOPEN,<lunghezza record RAM>

...fissa la lunghezza di tutti i record, inizializza il record attuale a 0, ma NON pulisce la memoria.

Il comando:

BANKWRITE,@<codice di uscita>, <stringa>[, <Numero record RAM>]

...scrive l'<espressione stringa> nel file RAM.

Il <numero record RAM> specifica quale record deve essere scritto. Se il parametro viene omesso viene usato il numero di record attuale. Il numero di record attuale viene poi fatto puntare a quello seguente.

Se l'<espressione stringa> non riempie completamente il record, i vecchi caratteri (che non sono stati sovrascritti) resteranno alla fine del record. Se l'<espressione stringa> è più lunga del record, i caratteri eccedenti verranno ignorati per evitare inserimenti nel record successivo.

Il <codice di uscita> è una variabile intera che restituisce il numero del record scritto (se l'operazione si è conclusa con successo), o un numero negativo se l'operazione di scrittura non ha avuto esito.

-1 Errore di fine file. Il numero di record richiesto eccede i 64K

-3 Errore di selezione del banco. (Non deve mai accadere).

Esempi:

```
|BANKOPEN,10
|BANKWRITE,@r%,"123testing",0
|BANKWRITE,@r%,w$
```

Il comando:

|BANREAD,@<codice di uscita>,@<stringa variabile>[,<numero record RAM>]

...legge un record dal file RAM e lo pone nella <stringa variabile>.

Il <numero record RAM> specifica quale record deve essere letto. Se il parametro viene omesso viene usato il numero di record attuale. Il numero di record attuale viene poi fatto puntare a quello seguente.

Se i contenuti del record non riempiono completamente la <stringa variabile> i vecchi caratteri (che non sono stati sovrascritti) resteranno alla fine della <stringa variabile>. Se il record è più lungo della <stringa variabile>, i caratteri eccedenti verranno ignorati, in quanto è impossibile aumentare la lunghezza della stringa variabile in un comando esterno.

Il <codice di uscita> è una variabile intera che restituisce il numero del record letto (se l'operazione si è conclusa con successo), un numero negativo se l'operazione di lettura non ha avuto esito.

-1 Errore di fine file. Il numero di record richiesto eccede i 64K

-3 Errore di selezione del banco. (Non deve mai accadere).

Esempi:

```
|BANKREAD,@r%,i$,0
```

Ricerca...

E' possibile ricercare, nei record memorizzati, una voce particolare.

Il comando:

|BANKFIND,@<codice di uscita>,<stringa da cercare>[,<numero record iniziale>[,<numero record finale>]]

...cercherà la stringa in tutti i record RAM. Il <numero record iniziale> specifica da quale record iniziare la ricerca. Se tale parametro viene omesso viene usato il numero di record attuale.

La ricerca procederà, con passo <lunghezza record RAM> in tutto il secondo lotto di 64K fino a quando non si è trovata la voce cercata.

Se si specifica il <numero record finale> la ricerca terminerà dopo aver confrontato tale record (a meno che non sia stato precedentemente trovato).

Se la ricerca termina con successo il numero di record attuale viene fatto puntare nel record in cui è stata trovata la voce.

Il <codice di uscita> è una variabile intera che restituisce il numero del record dove è stata trovata la voce (se l'operazione si è conclusa con successo), o un numero negativo se l'operazione non ha avuto esito.

-1 Errore di fine file. Il numero di record iniziale o quello finale eccede i 64K

-2 Non è stata trovata alcuna voce

-3 Errore di selezione del banco. (Non deve mai accadere).

La <stringa da cercare> può contenere caratteri jolly, ovvero CHR$(0) e il confronto viene fatto facendo riferimento o alla <lunghezza del record RAM> o alla lunghezza della <stringa cercata>, comunque sia corta.

Esempi:

```
|BANKFIND,@r%,"123test",0
|BANKFIND,@r%,f$,100,200
```

Attenzione ai confronti...

Errori ovvi, come il numero di parametri, vengono segnalati come errori 'Bad Command' (Comando errato). Comunque il meccanismo dei comandi esterni non scopre errori di tipo 'Type mismatch' e l'utente deve assicurarsi che si sia impiegata la forma di parametri corretta.

Il programma che segue usa i comandi del disco RAM per fissare ed interrogare un archivio contenente anagrammi di parole formate da 7 lettere. Esso cerca la voce richiesta ed è possibile usare i caratteri jolly.

Ad esempio, gli anagrammi della parola FIGURES che potrebbe essere confrontato con ?RUGS?? (gli ultimi due ?? possono essere tolti, se si desidera) sono FRUGSIE, FRUGSEI, IRUGSFE,IRUGSEF,ERUGSFI e IRUGSIF.

L'archivio richiede del tempo per essere creato, ma 64K è molta memoria da riempire!

```
10 'ANAGRAMMI di ROLAND PERRY
20 ' copyright (c) AMSOFT 1985
30 '
40 'Si ricordi di eseguire "BANKMAN" prima di eseguire il programma
60 '
70 MODE 2
80 DEFINT a-z
90 r%=0:|BANKOPEN,7
100 INPUT "Quale parola di 7 lettere volete gli anagrammi?;s$
110 IF LEN/s$)<>7 THEN 100
120 PRINT "Attendere per favore..."
130 LOCATE 1,5:PRINT"Sto calcolando:"
140 FOR c1=1 TO 7
150 FOR c2=1 TO 7
160 IF c2=c1 THEN 370
170 FOR c3=1 TO 7
180 IF c3=c2 OR c3=c1 THEN 360
190 FOR c4=1 TO 7
200 IF c4=c3 OR c4=c2 ORc4=c1 THEN 350
210 FOR c5=1 TO 7
220 IF c5=c4 OR c5=c3 OR c5=c2 OR c5=c1 THEN 340
230 FOR c6=1 TO 7
240 IF c6=c5 OR c6=c4 OR c6=c3 OR c6=c2 OR c6=c1 THEN 330
250 FOR c7=1 TO 7
260 IF c7=c6 OR c7=c5 OR c7=c4 OR c7=c3 OR c7=c2 OR c7=c1 THEN 320
```

Continua nella pagina seguente

```
270 o$=MID$(s$,c1,1)+MID$(s$,c2,1)+MID$(s$,c3,1)+MID$(s$,c4,1)
+MID$(s$,c5,
1)+MID$(s$,c6,1)+MID$(s$,c7,1)
280 LOCATE 12,5: PRINT x;o$
290 |BANKWRITE,@r%,o$
300 IF r%<0 THEN STOP
310 x=x+1
320 NEXT c7
330 NEXT c6
340 NEXT c5
350 NEXT c4
360 NEXT c3
370 NEXT c2
380 NEXT c1
390 lastrec=r%
400 REM ora guardiamoli
410 r%=0:g$=SPACE$(7)
420 PRINT:INPUT"Quale confronto vuoi fare: usa ? come carattere jolly: ",m$
430 m$=LEFT$(m$,7)
440 FOR x=1 TO LEN(m$)
450 IF MID$(m$,x,1)="?" THEN MID$(m$,x,1)=CHR$(0)
460 NEXT
470 |BANKFIND,@r%,m$m0,lastrec
480 IF r%<0 THEN GOTO 420
490 |BANKREAD,@r%,g$
500 PRINT g$,
510 |BANKFIND,@r£,m$mr%+1,lastrec
520 GOTO 480
```

Capitolo 8
A vostra disposizione

Questo capitolo permette di illustrare alcuni concetti del mondo dei calcolatori in generale e del 464/6128 in particolare. Non è necessario leggerlo prima di utilizzare il computer ma può aiutare a capire ciò che sta succedendo.

Parte 1: In senso generale ...

L'Arte del possibile

Anche se si è acquistato il 464/6128 solo per giocare con i giochi disponibili, si potrebbe voler apprendere ulteriori aspetti su ciò che viene definito 'hardware'.

L'hardware è tutto ciò che si può prendere in mano e spostare: la tastiera del computer, il monitor, le connessioni, ecc. In effetti è tutto ciò che non viene chiamato software - programmi, manuali, informazioni contenute su cassette o dischi.

Alcuni modi di comportamento del computer vengono permessi dalle possibilità dell'hardware, si pensi ad esempio al colore di un televisore o del monitor; è inoltre il software che utilizza l'hardware in modo da produrre caratteri e disegni sullo schermo.

E' l'hardware che conduce il raggio di elettroni sulla superficie luminescente dello schermo o del televisore; il software aggiunge ordine ed intelligenza indicando all'hardware quello che deve fare o come deve farlo. Aggiunge sincronizzazione, controllo e sequenza al fine ad esempio di produrre una astronave in partenza o qualcosa di più terrestre come i caratteri che appaiono sullo schermo quando vengono premuti determinati tasti.

Cosa rende un computer migliore di un altro?

L'hardware senza il software non ha valore. Il software senza l'hardware è anch'esso senza valore: il vero valore di un computer si dimostra quando entrambi interagiscono. Riguardo alla potenza di un calcolatore si possono fare alcune considerazioni fondamentali.

Le caratteristiche principali di un personal computer sono:

1. Risoluzione dello schermo - il più piccolo oggetto che è possibile riconoscere sullo schermo.

Questa è una combinazione di fattori, incluso il numero di colori diversi disponibili per il programmatore, il numero di aree che possono essere usate per la visualizzazione (i pixel) e il numero di caratteri di testo che possono essere visualizzati su uno schermo.

Si troverà che il 464/6128 è eccellente, se lo si paragona con altri computer di prezzo simile.

2. Interprete BASIC

Tutti i personal computer, praticamente, includono un interprete BASIC che permette all'utente di creare programmi ed usare le caratteristiche hardware. Il linguaggio di programmazione incluso (BASIC) nel 464/6128 è esso stesso un programma complicato che è stato sviluppato da molte persone negli Stati Uniti. Il 'Beginners All-purpose Symbolic Instruction Code' (Codice di istruzioni simboliche di uso generale per principianti) è il linguaggio maggiormente usato nel mondo, e come tutti i linguaggi ha diversi 'dialetti'.

La versione del 464/6128 è una delle più compatibili e potrà eseguire molti dei programmi scritti per il sistema operativo CP/M. E' una velocissima implementazione del BASIC, esegue in altre parole calcoli in maniera velocissima: un computer può impiegare 0.005 secondi per moltiplicare 3 per 5 mentre un altro può impiegarne 0.075; un programma che disegna grafici può impiegare 0.05 secondi per ripetere cento volte un calcolo ripetitivo, un altro computer potrebbe impiegarne 0.075: pochi secondi in più costituiscono una considerevole differenza di prestazioni.

Si potrà spesso sentir parlare di 'codice macchina'. Il codice macchina è una forma di istruzione in codice che viene inviata al processore. Questa richiede meno tempo per essere elaborata e permette di aumentare da 5 fino 15 volte la velocità rispetto ad un normale interprete BASIC. Ma occorre un tempo superiore dalle 5 alle 50 volte per scrivere un programma in codice macchina invece che in BASIC.

Il BASIC AMSTRAD è tra i più veloci e ben dotati: incorpora molte caratteristiche utili per gli esperti di programmazione BASIC che potranno superare la lentezza dell'interprete di un linguaggio ad alto livello per ottenere effetti visivi e sonori sorprendenti.

3. Espandibilità

Molti computer hanno bisogno di ulteriori strumenti hardware: stampanti, joystick, dischi. Paradossalmente la maggior parte dei computer domestici più di successo hanno bisogno di 'interfacce di espansione' prima di poter installare una stampante o un joystick.

L'acquirente non sempre pensa all'immediato futuro, in quanto una macchina che include una porta per stampante parallela (Centronics compatibile) e per joystick può essere, alla fine dei conti, meno cara.

Tra le caratteristiche del 464/6128 va inclusa la porta Centronics, una porta per un secondo disco (solo 6128), la possibilità di aggiungere due joystick, una uscita del suono stereo, ecc, oltre ad un bus di espansione che può essere usato per collegare una interfaccia seriale (RS232C), un MODEM, un sintetizzatore vocale, ecc.

4. Suono

Le caratteristiche sonore di un computer determinano se il computer suona in maniera sorda o produce una accettabile rappresentazione degli strumenti musicali.

Il 464/6128 utilizza un generatore sonoro a tre canali e 8 ottave che può produrre una qualità musicale accettabile con un controllo completo degli inviluppi di ampiezza e tono. Inoltre, il suono è in stereo dove un canale fornisce l'uscita sinistra e l'altro la destra: il terzo canale si trova nel mezzo.

Ciò fornisce una vasta gamma di possibilità di effetti sonori quando ad esempio si scrivono programmi di giochi.

Abbiamo terminato di fornire una idea delle caratteristiche: deciderà l'utente se sono importanti per lui. Speriamo comunque, che l'utente le voglia provare ed ottenere il massimo dal computer.

Cosa posso fare?

Con tutte le potenti tecnologie elettroniche gli utenti si chiedono frequentemente come mai anche una macchina avanzata come il 464/6128 non può apparentemente visualizzare ad esempio i tipi di immagini che siamo abituati a vedere su uno schermo televisivo. Perché un computer non può animare una figura, come ad esempio un uomo che cammina, sullo schermo in modo un po' più naturale? Perché tutti i computer rappresentano i movimenti con figure che sembrano dei 'robot'?

La risposta è complessa. Una risposta semplicistica sarebbe quella di dire che lo schermo di un computer non ha nulla a che fare con lo schermo di un televisore. Un televisione opera usando informazioni lineari che descrivono un'area infinita di risoluzioni tra gli estremi 'luce' e 'scuro' di tutti i colori di uno spettro. Questo processo, in termini del computer, implica che la 'memoria' di visualizzazione di una figura sul televisore è 20 volte maggiore se paragonata con una analoga sul video del computer.

Questa è solo una parte del problema, poichè l'animazione di una figura richiede una enorme quantità di memoria che deve essere elaborata ad una velocità elevata (50 volte al secondo). Tutto ciò può essere svolto da macchine che costano molti milioni in più di un normale personal computer!

Fino a quando il costo della memoria non crollerà i piccoli computer dovranno avere a che fare con quantità di memoria relativamente piccole per controllare lo schermo: ciò comporterà una bassa risoluzione e movimenti 'goffi'. Sebbene un hardware ottimamente progettato possa fare il meglio non potrà comunque competere con i computer che possono produrre movimenti del tipo dei cartoni animati.

Questa tastiera mi sembra di conoscerla ...

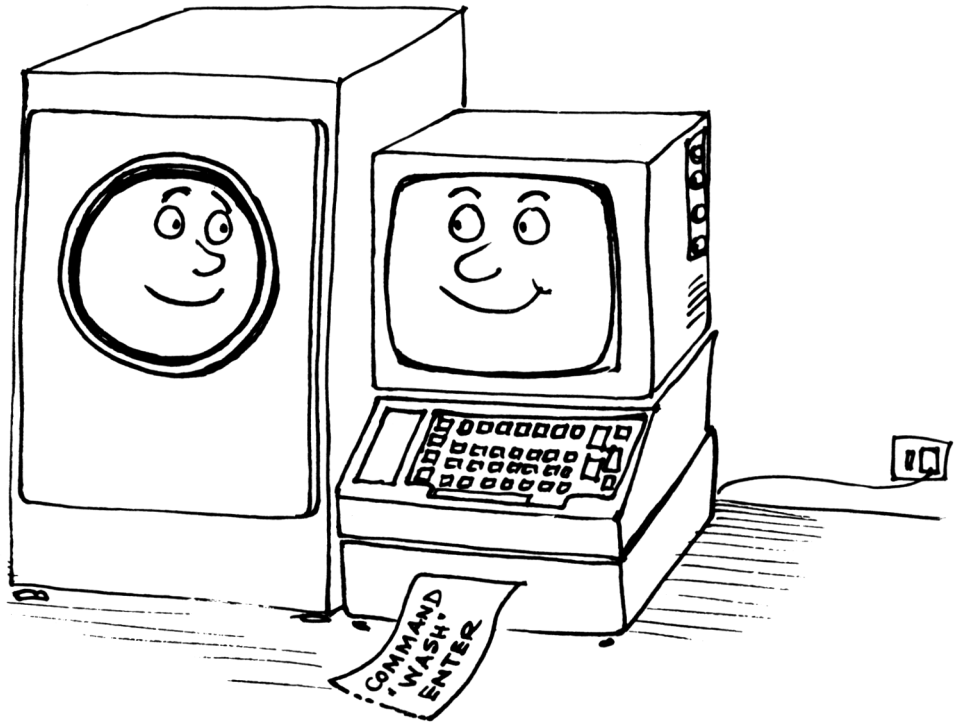
Perchè non si può accendere il computer e scrivere immediatamente e semplicemente una pagina di testo?

Non ci si faccia ingannare dal fatto che la tastiera del computer assomiglia a quella di una macchina per scrivere. Lo schermo non è un foglio elettronico, è una console di comando: questo significa che aiuta a far comunicare l'utente con la memoria della macchina mediante un linguaggio di programmazione.

Fino a quando non gli si indichi di fare diversamente, il computer cercherà di eseguire tutto quello che viene immesso alla tastiera come se fosse una istruzione di un programma. Quando si preme il tasto **[RETURN]** il computer cercherà di scoprire se in ciò che si è inserito ci sia qualche cosa che abbia senso per BASIC in caso contrario commenterà:

```
Syntax error
```

Comunque potrebbe accadere che il programma residente in memoria è in realtà un elaboratore di testi, in tal caso è possibile inserire parole a caso, premere **[RETURN]** e scrivere tutto ciò che si vuole come se lo si facesse su una macchina per scrivere. Ma per far ciò si deve aver caricato in memoria un programma di elaboratore testi.

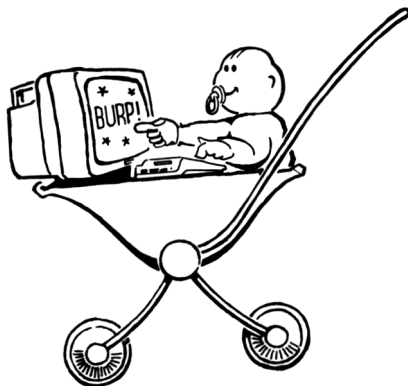
Un computer 'sembra' unire insieme diversi oggetti di uso quotidiano come il televisore e la macchina per scrivere: occorre ricordarsi comunque che queste similitudini sono superficiali e che il computer ha una personalità completamente diversa.

Chi ha paura del “gergo”?

Così come tutte le discipline, il mondo dei calcolatori ha sviluppato un proprio linguaggio come forma di abbreviazione per concetti complicati che richiedono più parole per essere spiegati. Non è solo l’alta tecnologia colpevole di nascondersi dietro una apparente cortina di fumo o ‘ronzii di parole’, gergo e terminologia; quanti di noi si sono scontrati con le barriere della comprensione, che esistono in tutti i campi della vita?.

La maggiore differenza tra i gerghi sorge dal modo con cui le parole vengono usate piuttosto che dal loro significato stesso. La maggior parte delle persone che diventano familiari con la terminologia del calcolo adotterà un modo personale di uso delle parole, il più diretto possibile, al fine di minimizzare la complessità della comunicazione.

Non bisogna sentirsi ingannati da tale ‘linguaggio chiaro’ usato nel calcolo, non è un oggetto letterale ma una scienza precisa, separata dalla ‘sintassi’ delle parole; la struttura della comunicazione è molto diretta, e non confusa o ambigua. Gli insegnanti di informatica non hanno cercato di rendere ciò una forma d’arte cercando di analizzare il significato esatto di una parola secondo quello che intende un programmatore nella costruzione di un programma.

Avendo detto ciò, sia che il significato di programma sia o meno noto, vi sono ancora diversi aspetti che possono essere analizzati sia in modo elegante che disordinato, ed è stata data più importanza ad un approccio formale nella costruzione di un programma ora che il danno inizialmente portato dalla micro rivoluzione si è stabilizzato.

Il calcolo viene appreso rapidamente da molte persone giovani che apprezzano la precisione e la semplicità delle idee ed il modo con cui queste possono essere comunicate; non si troveranno molti ragazzi di dieci anni avvocati, ma sarà possibile trovare molti programmatori!

Le basi del BASIC

Quasi tutti i computer domestici forniscono un linguaggio noto come BASIC il quale permette di ottenere programmi scritti nel modo migliore e con un linguaggio semplice e subito disponibile. BASIC è un abbreviazione di 'Beginners All-purpose Symbolic Instruction Code': molti programmi estremamente complessi e potenti sono scritti usando BASIC.

Comunque non c'è dubbio che il nome ha attratto molti principianti per la promessa di fornire un punto di partenza nel labirinto dei linguaggi di programmazione e ha contribuito notevolmente alla sua divulgazione.

BASIC è un linguaggio di programmazione che interpreta una gamma di comandi permessi ed esegue le operazioni sui dati in fase di esecuzione del programma. Diversamente dal vocabolario umano che comprende dalle 5000 alle 8000 parole (oltre alle varie desinenze dei verbi che si possono usare, ecc) il BASIC ne possiede duecento. I programmi scritti usando BASIC devono seguire regole rigide riguardanti l'uso di tali parole. La sintassi è precisa e qualsiasi tentativo di comunicazione con il computer mediante l'uso di espressioni letterali o colloquiali darà come risultato un freddo messaggio:

```
Syntax error
```

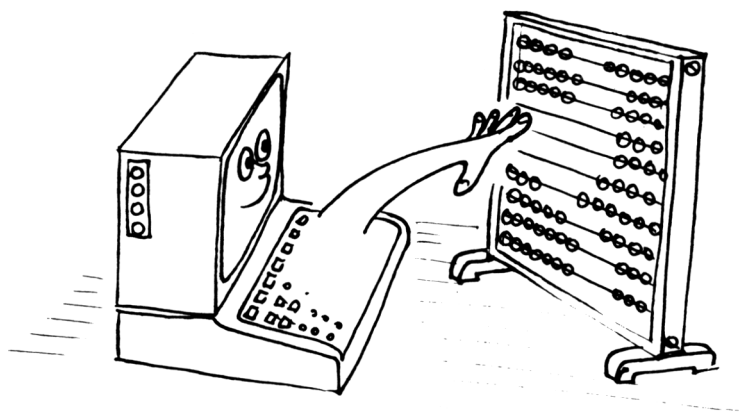
ovvero errore di sintassi.

Ciò non è restrittivo come potrebbe sembrare ad un primo momento poichè il linguaggio BASIC (la sua sintassi) è progettato prima di tutto per trattare numeri, dati numerici. Le parole sono essenzialmente una estensione dei familiari operatori aritmetici */- ecc, e il concetto più importante da comprendere dai principianti è quello che un computer lavora solo con un dato numerico: informazione che è fornita al Processore Centrale (circuito integrato) solo sotto forma di dati numerici.

Numeri per favore!

Se un computer viene usato per memorizzare tutto il lavoro di Shakespeare, nel sistema non si troverà una sola lettera o parola. Ogni informazione viene prima convertita in un numero che il computer può localizzare e successivamente trattare come richiesto.

BASIC interpreta le parole come numeri che il computer può successivamente trattare usando solo l'addizione, la sottrazione ed alcune caratteristiche Booleane che permettono di confrontare dati e selezionarne altri in base a determinati attributi: cioè controllare se un numero sia più grande o uguale ad un altro, o eseguire un particolare compito se un numero 0 un altro incontra una certa scelta.

Mediante il programma il computer suddivide ogni compito in una semplice serie di operazioni che implicano due sole risposte : SI o NO.

Che tale processo sembri scomodo non è del tutto sbagliato, poichè si è appena scoperta una delle più importanti verità riguardo il calcolo. Un computer è prima di tutto uno strumento per l'esecuzione di compiti semplici e ripetitivi in modo veloce e con precisione assoluta. Così BASIC interpreta le istruzioni fornite sotto forma di programma e le traduce nel linguaggio che può essere usato dalla CPU. Solo due stati vengono compresi dalla logica del computer: 'si' o 'no' rappresentati in notazione binaria da '1' e '0'. La rappresentazione Booleana è semplicemente 'vero' e 'falso', non vi sono espressioni come 'possibile' o 'forse'.

Il processo di commutazione tra questi due stati distinti è l'essenza del termine digitale. Nella natura la maggior parte dei processi si muovono gradualmente da uno stato completamente 'stabile' in un altro in una progressione omogenea. In altre parole la transizione è fatta seguendo il percorso di una linea tra due stati; in un ambiente digitale ideale la commutazione da uno stato al seguente avviene istantaneamente, ma la fisica dei semiconduttori porterà un ritardo a cui si fa riferimento come ritardo di propagazione; è l'accumulazione di molti di questi ritardi la ragione per cui il computer impiega del tempo per fornire una informazione.

In qualsiasi caso, il computer deve attendere che un compito venga finito prima che un secondo possa usare il risultato del primo: quindi ci dovrebbe essere comunque un ritardo imposto. Il processo digitale è nero o bianco dove i periodi nella transizione mediante sfumature di grigi non hanno significato. La progressione analogica o lineare utilizza varie sfumature di grigio.

Se in definitiva la risposta è 0 o 1, non vi è possibilità che sia 'quasi' corretta. In effetti può sembrare a volte che il computer faccia degli errori nel trattamento dei dati numerici a causa della limitazione della dimensione dei numeri che esso può utilizzare; in altre parole, numeri molto grossi o molto piccoli dovranno essere ristretti all'interno dello spazio standard che è stato loro assegnato e ciò porta ad errori di arrotondamento per cui 999.999.999 può diventare 1.000.000.000.

In altre parole dove vi sono solo due numeri disponibili, 0 e 1, come si può contare dopo l'uno?

Bit e byte

Abbiamo appena accennato all'uso dei numeri basati sul sistema decimale nel quale il punto di riferimento è 10, vi sono cioè dieci cifre disponibili per rappresentare quantità le quali vanno da 0 a 9 (che sono prevalentemente usate da 1 a 10). Il sistema in cui la gamma di numeri va da 0 a 1 è il sistema binario e le cifre del sistema sono dette bit, una abbreviazione di 'BInary digiT' (cifra binaria).

La relazione tra bit e notazione decimale è semplice da comprendere:

E' convenzione attuale dichiarare il numero massimo di cifre binarie usate aggiungendo all'inizio del numero alcuni zeri

7 decimale diventa
00111 binario
usando 5 bit di notazione

Nel sistema binario, le cifre possono essere considerate semplicemente come indicatori in colonne per specificare se deve o meno venir fornita la potenza di due

$2^0 = 1$
$2^1 = 2 = 2 = 2(2^0)$
$2^2 = 4 = 2x2 = 2(2^1)$
$2^3 = 8 = 2x2x2 = 2(2^2)$
$2^4 = 16 = 2x2x2x2 = 2(2^3)$

quindi la colonna apparirà

2^4	2^3	2^2	2^1	2^0
1	0	0	1	1
(16	0	0	2	1) = 19

Al fine di fornire un metodo per abbreviare il riferimento a una in cifra binaria, viene usato il termine byte che denota 8 bit di informazione. Il numero più alto che può essere memorizzato in un byte è (binario) 11111111 o (decimale) 255. Ciò implica 256 variazioni incluso 00000000, che è ancora un dato valido per un computer.

I computer tendono a trattare dati in multipli di 8 bit. 256 non è un numero molto grande. Quindi, per ottenere un metodo accettabile per gestire la memoria, si usano due byte al fine di fornire un metodo di indirizzamento della memoria sotto forma di vettore, con un indirizzo orizzontale ed uno verticale mediante il quale gli elementi del vettore possono essere localizzati:

	0	1	2	3	4	5	6	7	8	9
0										
1										
2										
3										
4										
5				1		1				
6										
7										
8										
9										

Questo vettore può contenere (10x10) oggetti di informazione usando i numeri di indirizzo compresi nel campo 1 - 9. L'oggetto memorizzato nella posizione 3,5 è '1', così come l'oggetto nella posizione 5,5.

Quindi un vettore binario di 256x256 può trattare 65.536 locazioni usando un indirizzo ad 8 bit per gli assi verticali e orizzontali del vettore. Quindi il nostro '0' e il nostro '1' sono progrediti rendendosi in grado di identificare uno dei 65.536 elementi.

Il livello successivo di abbreviazione in notazione binaria è kilobyte (kByte o semplicemente 'K') il quale è formato da 1024 byte. 1024 è il multiplo binario più vicino alla nostra notazione di 'chilo' e spiega perchè un computer descritto '64K' di memoria ha in effetti una memoria di 65.536 byte (64x1024).

Grazie all'interprete BASIC non è necessario effettuare conversioni ed è possibile essere dei bravi programmatori senza conoscere il sistema binario. Ma una valutazione del significato binario aiuterà a distinguere i molti 'magici' o significativi numeri che inevitabilmente salteranno fuori nel corso del lavoro.

E' bene apprendere il sistema binario e i vari significati dei numeri 255, 1024, ecc. poichè è molto improbabile che questi cambieranno in futuro. La certezza e la semplicità derivante dal lavoro con solo due stati prevarrà notevolmente sulla complessità derivante dall'uso di altri numeri base.

Comunque...

Semplice o elegante che sia, il sistema binario è prolisso e comporta imprecisioni poichè non può esssere letto facilmente con una occhiata. Il codice binario ha un certo numero di codici associati che agiscono come abbreviazione per i programmatori. Uno di tali codici, ampiamente usato nei microcomputer, è detto Esa (abbreviazione di esadecimale).

Il numero è basato sul 16 ed è rappresentato da un solo carattere:

Decimale

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Esa

0 1 2 3 4 5 6 7 8 9 A B C D E F

Il sistema esadecimale può suddividere gli otto bit di un byte in due blocchi di quattro bit poichè 15 è il numero a quattro bit: 1111 binario. Il primo blocco indica il numero di unità di '15' complete, il secondo indica il resto; è in questo caso che emerge l'eleganza del sistema binario e delsistema esadecimali.

Riconsideriamo la tabella introdotta nella notazione binaria.

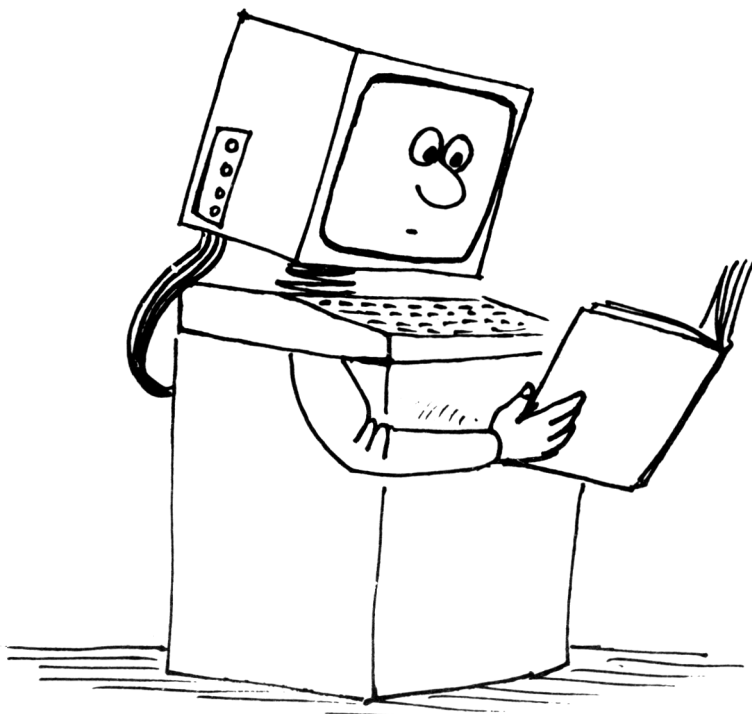
Decimale	Binario	Esadecimale
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

Il numero a 8 bit 11010110 può essere suddiviso e considerato come due numeri a 4 bit noti come nibbles, Hex D6. In questa guida un numero basato su notazione esadecimale verrà introdotto con un simbolo '&' cioè &D6, questo è il numero maggiormente usato dai programmatori che usano tecniche di linguaggio assembler. Un programma in tale linguaggio è più vicino alla programmazione in codice macchina in quanto tale linguaggio permette di usare semplici lettere 'mnemoniche' per specificare i 'numeri' effettivi di codice macchina.

Quando si usano numeri esadecimali, occorre prima di tutto calcolare il valore della prima cifra per ottenere il numero di "16" contenuti nel numero finale, e successivamente aggiungere la seconda 'metà' della notazione esadecimale per ottenere il numero decimale equivalente. Si può essere tentati di considerare il numero &D6 come 13+6, o 136. Ma è (13x16)+6=214

E' lo stesso processo che si usa quando si legge un numero decimale come '89', ovvero (8x10)+(9). Moltiplicare per dieci è notevolmente più semplice a meno che non si è fatta molta pratica moltiplicando per 16.

Se si è appreso tutto ciò senza fare troppa confusione allora si è in grado di apprendere le basi del calcolo. Ci si potrebbe essere resi conto che tutto lo scalpore è qui: il che non è del tutto sbagliato. Il computer è un gestore che dirige in modo semplice concetti ed idee: si possono eseguire compiti ad una velocità elevata (milioni di volte per secondo) e con una smisurata capacità di ricordare sia i dati inseriti che i risultati intermedi delle migliaia di semplici somme che producono il risultato.

Se si desidera imparare di più sulla teoria del mondo dei computer vi sono centinaia di libri disponibili su tale soggetto. Alcuni tendono a provocare più confusione di quella che si aveva all'inizio di tale lettura e pochi rivelano realmente la semplicità e la relazione esistente tra i sistemi numerici ed il modo con cui il computer li tratta.

Parte 2: Qualcosa di più sul 464/6128

Questa parte spiega alcuni aspetti più specifici dei 464/6128. Le informazioni necessarie per comprendere questi argomenti sono state presentate nel corso introduttivo e nel capitolo che riporta l'elenco completo delle parole chiave del BASIC del 464/6128.

Argomenti trattati in questa parte:

* Set di caratteri
* ASCII
* Variabili
* Logica
* Caratteri definibili dall'utente
* Formattazione della stampa
* Finestre
* Interruzioni (interrupts)
* Dati
* Suono
* Grafica
* Grafica utilizzando la seconda parte della memoria

Un po' di caratteri

Premendo i tasti del 464/6128 non dovreste essere sicuri che ciò che apparirà sullo schermo siano necessariamente numeri e lettere riconoscibili. Dopotutto abbiamo appena detto che il computer non è una macchina per scrivere. Ciò che accade effettivamente è il risultato della pressione di interruttori elettrici. I segnali elettrici prodotti premendo tali interruttori vengono tradotti dai circuiti che compongono la macchina, i quali a loro volta producono un insieme di punti sullo schermo. Noi riconosciamo questi gruppi di punti come lettere, numeri o altri caratteri del set di caratteri.

Alcuni dei caratteri non sono accessibili direttamente mediante la pressione di un tasto ma possono essere visualizzati usando l'istruzione PRINT CHR$(<numero>). Ciò avviene Perché ogni elemento contenuto nella memoria del computer è contenuto in una unità chiamata byte e, come abbiamo già visto nella parte 1 di questo Capitolo, un byte può contenere 256 diverse combinazioni di valori. Dal momento che il computer deve utilizzare almeno un byte per ogni carattere che può utilizzare (che lo si voglia o no, questa è la parte più piccola che il 464/6128 può riconoscere), potremo usare tutte le 256 possibili combinazioni invece di doverci accontentare dei circa 96 caratteri standard che possono essere stampati da una normale macchina per scrivere.

Tali caratteri standard costituiscono solo un sottoinsieme dei caratteri generabili. Questo viene indicato come il sistema ASCII, un termine derivato da "American Standard Code for Information Interchange", un sistema di codifica che fa in modo che i caratteri inviati da un computer possano essere riconosciuti da un computer che riceve dal primo dei dati. Esiste nel manuale un capitolo di riferimento contenente tutti i caratteri ASCII e tutti i caratteri aggiuntivi utilizzati dal 464/6128, insieme ai corrispondenti codici.

Come mai abbiamo parlato di queste cose ...

Probabilmente si sarà già visto questo programma:

```
10 FOR n=32 TO 255
20 PRINT CHR$(n);
30 NEXT
```

... che chiede al computer di visualizzare tutti i caratteri utilizzabili. Passiamo ora ad esaminare il programma:

La prima cosa da notare è che non si è chiesto al computer di stampare i caratteri con una istruzione come PRINT "abcdefghijklm.....ecc" ma PRINT CHR$(n). n rappresenta semplicemente una variabile; la scelta della lettera è arbitraria e la n è semplicemente la preferita dai matematici in questo campo. Una variabile è semplicemente un oggetto che contiene una informazione che può variare in dipendenza dalle istruzioni che compongono il programma.

Che cos'è una variabile ?

Un numero come 5 è fissato e si trova tra il 4 ed il 6 e quindi non è una variabile; il carattere n è anch'esso fissato (è una lettera dell'alfabeto).

Quindi come fà il computer a distinguerle? Se la lettera n è intesa come carattere alfabetico, dobbiamo racchiuderla tra virgolette come in "n", ed il computer avrebbe risposto con il messaggio Syntax error in quanto non può comprendere la sequenza FOR "n"=32 TO 255.

Usando la n come abbiamo fatto, abbiamo detto al computer che n è una variabile. Per definizione, in BASIC il comando FOR deve essere seguito dal nome di una variabile e quindi il computer assume che qualunque cosa segua un FOR lo sia.

Abbiamo anche detto al computer n=32 to 255. Abbiamo così stabilito che la variabile può assumere valori in sequenza tra 32 e 255.

Dopo aver dichiarato questa variabile, dobbiamo dire al computer cosa farne e la linea successiva fà proprio questo:

```
20 PRINT CHR$(n);
```

Questa linea dice al computer di convertire il valore numerico assegnato ad n nel carattere con il numero corrispondente e di stamparlo.

Il punto e virgola alla fine di questa riga, indica al computer di non andare a capo come accadrebbe normalmente in modo che i caratteri vengano stampati uno in fianco all'altro e non uno sotto l'altro.

La linea successiva indica al computer che, quando ha effettuato l'operazione con il primo numero della sequenza, deve tornare alla linea dove si trova il FOR ed effettuare la stessa operazione con il prossimo (NEXT) valore assegnato alla variabile n. Questo processo è noto con il nome di ciclo ed è uno degli aspetti più importanti dell'utilizzo e della programmazione dei computer. Esso permette di evitare di immettere lunghe e ripetitive sequenze di comandi e si imparerà presto ad utilizzarlo nei propri programmi.

Quando il ciclo FOR raggiunge il limite specificato (255), l'operazione termina ed il computer prosegue l'esecuzione con la prima linea dopo la linea 30; ma non ve sono e dunque viene terminata l'esecuzione e viene ripresentato il prompt Ready. Il prompt indica che il computer è pronto (ready) ad accettare ulteriori informazioni; è anche possibile immettere nuovamente un RUN e ripetere l'esecuzione del programma. Il programma è memorizzato al sicuro nel computer e rimarrà là finché non si dice al computer di memorizzarlo altrove o finché non si spegne il computer.

Questo programma illustra un aspetto fondamentale dei computer: tutto quello che essi fanno è in relazione con i numeri. Il computer ha visualizzato l'alfabeto (insieme ad altri caratteri) usando un numero per identificare ciascun carattere. Quando si preme il tasto **A,** non si chiede al computer di stampare una A sullo schermo ma di ricercare nella sua memoria le informazioni necessarie a stampare la lettera A sullo schermo. L'effettiva posizione di questo dato è definita dal codice numerico attivato dalla pressione del tasto.

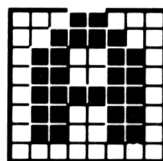
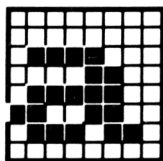
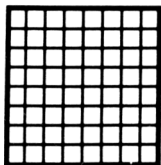
Ogni carattere corrisponde ad un numero; questi sono elencati nella Parte 3 del Capitolo di riferimento.

In modo simile i caratteri visualizzati non hanno nulla a che vedere con la scrittura della lettera sullo schermo, ancora una volta si tratta solamente di numeri.

Ad esempio, il codice della lettera A è 97. Il computer non può comprendere il numero 97 così com'è e quindi il numero deve essere convertito in un formato gestibile dal computer: il codice macchina. I principi riguardanti quest'aspetto della macchina si sono discussi precedentemente in questo Capitolo.

Al primo impatto, la traduzione dal codice decimale a quello esadecimale potrà risultare piuttosto difficile. Pensare a numeri basati su dieci unità è così naturale che fare in un altro modo è come cercare di mangiare con il coltello e la forchetta scambiati nelle mani.

Occorre un buon grado di scaltrezza mentale per comprendere la notazione esadecimale; una volta compresa questa, molte cose riguardanti i calcolatori diverranno più chiare ed apparirà in modo più chiaro la struttura del sistema di numerazione.

Una volta che il computer ha convertito la pressione del tasto **A** nel tipo di numeri da esso conosciuti ricerca nella parte di memoria indicata ed il risultato è un'altra serie di numeri che definiscono il carattere. In pratica il carattere visualizzato sullo schermo è costituito da un blocco di dati, memorizzato in forma di matrice numerica.

Matrice vuota di un carattere a minuscola A maiuscola.

Gli elementi della matrice sono righe e colonne di punti. Il carattere viene visualizzato accendendo o spegnendo i punti appropriati; ogni punto è determinato dai dati memorizzati nella memoria del computer. Vi sono 8 righe di 8 punti in ogni carattere del 464/6128 e, se non si trova il carattere desiderato nel set di 256 di cui è dotata la macchina, è possibile definirne di propri usando la parola chiave SYMBOL presentata più avanti in questo Capitolo.

I caratteri definibili dall'utente usano una qualunque combinazione dei 64 punti che costituiscono la matrice. Perciò, costruendo tutte le possibili combinazioni dei punti di un carattere, è possibile creare molti più elementi (caratteri). Si aggiunga a ciò il fatto che è possibile raggruppare diversi elementi per formare blocchi di maggiori dimensioni e si vedrà che le possibilità dei caratteri definibili dall'utente sono solo limitate dal tempo che si ha a disposizione e dal proprio ingegno.

Logica ...

La differenza principale tra una calcolatrice e il computer è la capacità di quest'ultimo di trattare operazioni logiche in applicazioni tipo sequenze condizionali IF...THEN. Per far ciò gli operatori logici trattano i valori a cui sono applicati come bit ed operano sul bit individuale. La descrizione e l'uso è totalmente logico: ma è notoriamente difficile descrivere la logica in termini semplici senza la precisione di definizioni concise.

Le due parti di una espressione logica sono note come argomenti. Una espressione logica comprende:

<argomento>[<operatore logico><argomento>]

dove:

<argomento> è: NOT <argomento>
oppure : <espressione numerica>
o : <espressione relazionale>
o : (<epressione logica>)

Entambi gli argomenti di un operatore logico sono costretti in una rappresentazione intera e se l'intero non è compreso nell'area ammessa viene visualizzato Error 6.

Gli operatori logici, in ordine di precedenza con cui vengono applicati ai bit sono:

AND restituisce 0 a meno che entrambi gli argomenti dei bit non siano 1
OR restituisce 1 a meno che entrambi gli argomenti dei bit siano 0
XOR restituisce 1 a meno che entrambi gli argomenti dei bit siano uguali

AND è l'operatore logico più usato:

```
PRINT 10 AND 10
```

... darà come risultato 10

```
PRINT 10 AND 12
```

Restituirà 8

PRINT 10 AND 1000

Restituirà ancora 8

Questo a causa del fatto che i numeri 10 e 1000 sono stati convertiti in rappresentazione binaria:

```
      1010
1111101000
```

L'operatore AND controlla ogni corrispondenza di bit alla volta e quando il bit in cima ed in fondo alla riga è 1 la risposta è 1.

0000001000

... il quale viene convertito in notazione decimale dà come risultato 8. Tutto ciò significa che l'operatore logico AND viene usato per scoprire se due condizioni si presentano simultaneamente. Ecco un programma autoesplicativo:

```
10 INPUT "Numero del giorno";giorno
20 INPUT "Numero del mese";mese
30 IF giorno=25 AND mese=12 THEN 50
40 CLS:GOTO 10
50 PRINT "Buon Natale!"
```

Anche OR opera sui bit, ma con esso il risultato è 1 a meno che entrambi i bit degli argomenti siano 0, nel qual caso il risultato è 0. Usando gli stessi numeri dell'esempio AND:

```
PRINT 1000 OR 10
1002
```

Bit a bit

```
      1010
1111101000
```

Fornisce come risposta:
1111101010

Ecco un programma di esempio:

```
10 CLS
20 INPUT "Numero del mese";mese
30 IF mese=12 OR mese=1 OR mese=2 THEN 50
40 GOTO 10
50 PRINT "Dev'essere inverno"
```

L'operatore NOT inverte ogni bit del suo argomento (0 diventa 1 e viceversa):

```
10 CLS
20 INPUT "Numero del mese";mese
30 IF NOT (mese=6 OR mese=7 OR mese=8) THEN 50
40 GOTO 10
50 PRINT "Non può essere estate!"
```

Un'altra caratteristica fondamentale è quella che consiste nell'unire insieme le diverse condizioni logiche:

```
10 INPUT "Numero del giorno"; giorno
20 INPUT "Il numero del mese"; mese
30 IF NOT(mese=12 OR mese=1) AND giorno=29 THEN  50
40 CLS:GOTO 10
50 PRINT "Questo non è nè Dicembre nè Gennaio ma potrebbe essere un anno
bisestile"
```

Il risultato dell'espressione relazionale è 1 o 0. La rappresentazione del bit per 1 è tutti i bit interi uguali a 1; per 0 tutti i bit di interi sono 0. Il risultato di una operazione logica su due di tali argomenti conterrà 1 per Vero o 0 per Falso.

Controlliamo ciò aggiungendo al programma la linea 60:

```
60 PRINT NOT(mese=12 OR mese=1)
70 PRINT (mese=12 OR mese=1)
```

Quando eseguiamo il programma inserendo 29 come giorno e 5 come mese si produrrà nella linea 50 la risposta e i valori effettivi verranno restituiti dalle espressioni logiche nelle linee 60 e 70.

Infine XOR (OR esclusivo) produce un risultato Vero se i suoi argomenti sono diversi.

La seguente tabella riassume tutte queste caratteristiche ed è nota come tabella di verità. E' un modo comodo per comprendere ciò che avviene nelle operazioni bit a bit.

Argomento A	1 0 1 0
Argomento B	0 1 1 0
risultato AND	0 0 1 0
risultato OR	1 1 1 0
risultato XOR	1 1 0 0

Caratteri definibili dall'utente

Uno dei primi utilizzi dei numeri binari che certamente si incontrerà, riguarda il disegno dei caratteri da utilizzare con il comando SYMBOL. Se il carattere viene disegnato in una griglia 8x8, ognuna delle 8 righe dovrà essere convertita in un numero binario inserendo un 1 per ogni pixel che dovrà essere colorato e 0 per ogni pixel trasparente (del colore della carta). Questi 8 valori vengono utilizzati dal comando SYMBOL. Ad esempio, per definire un carattere con una casetta:

```
    *      = 00001000   = &08   = 8                        = 8
  ****     = 00111100   = &3C   = 32+16+8+4                = 60
 *    *    = 01000010   = &42   = 64                    +2 = 66
* *  * *   = 10100101   = &A5   = 128  +32    +4      +1 = 165
*      *   = 10000001   = &81   = 128                   +1 = 129
* ** * *   = 10110101   = &B5   = 128  +32+16 +4      +1 = 181
* **   *   = 10110001   = &B1   = 128  +32+16         +1 = 177
********   = 11111111   = &FF   = 128+64+32+16+8+4+2 +1 = 255
```

.. il comando è:

```
SYMBOL 240,8,60,66,165,129,181,177,255
```

.. o ..

```
SYMBOL 240,&08,&3C,&42,&A5,&81,&B5,&B1,&FF
```

.. o ..

```
SYMBOL 240,&X00001000,   &X00111100,   &X01000010,   &X10100101,
           &X10000001,   &X10110101,   &X10110001,   &X11111111
```

Per stampare il carattere così definito si usa:

```
PRINT CHR$(240)
```

Infine, per raggruppare più caratteri, si può specificare

```
coppia$=CHR$(240)+CHR$(240)
PRINT coppia$
```

.. o ..

```
via$=STRING$(15,240)
PRINT via$
```

Altre notizie su PRINT

PRINT è uno dei primi comandi che si impara ad utilizzare. E' anche uno dei comandi BASIC che effettivamente fa ciò che dice, PRINT in inglese vuol dire stampa. Riguardo a PRINT vi sono però molte più considerazioni di quanto possa sembrare in un primo momento; ad esempio DOVE bisogna stampare? e COME bisogna stampare?

Formattazione della stampa

E' possibile usare il comando PRINT in vari modi. Il più semplice consiste nel farlo seguire dall'oggetto che deve essere stampato. Questo oggetto può essere un numero, una stringa o il nome di una variabile.

```
PRINT 3
 3

PRINT "ciao"
ciao

a=5
PRINT a
 5

a$="prova"
PRINT a$
prova
```

In uno stesso comando PRINT è possibile inserire più oggetti, separati da un separatore, da TAB o da SPC. I separatori utilizzabili sono la virgola ed il punto e virgola. Il punto e virgola causa la continuazione della stampa sulla stessa riga ed immediatamente dopo l'ultima posizione di stampa. La virgola provoca lo spostamento alla zona successiva. L'ampiezza iniziale della zona è 13, ma può essere cambiata usando il comando ZONE.

```
PRINT 3;-4;5
 3 -4  5

PRINT "ciao ";"ciao"
ciao ciao

PRINT "ciao","ciao"
ciao         ciao

PRINT 3,-4,5
 3           -4           5

ZONE 4
PRINT 3,-4,5
 3   -4   5
```

Si noti che i numeri positivi sono stampati con uno spazio davanti mentre i numeri negativi sono preceduti da un segno meno. Tutti i numeri hanno dunque un carattere davanti (spazio o meno). Le stringhe vengono stampate esattamente come appaiono all'interno delle virgolette.

La funzione SPC prende come parametro una espressione numerica e stampa altrettanti spazi. Se il numero è negativo, si assume zero. Se è più grande dell'ampiezza del canale (finestra), si assume l'ampiezza dello schermo.

```
PRINT SPC(5)"salve"
     salve

x=3
PRINT SPC(x*3)"salve"
         salve
```

TAB si comporta in modo molto simile ma stampa tanti spazi quanti ne occorrono per arrivare ad una determinata colonna.

Il canale in cui apparirà tutto ciò che viene stampato è la finestra 0, a meno che si sia incluso uno specificatore di canale prima degli oggetti da stampare. Per inviare dati ad un'altra finestra possono essere usati altri canali. I canali 8 e 9 sono casi speciali: il canale 8 è collegato ad una eventuale stampante: Il canale 9 è collegato al disco o alla cassetta. Tuttavia, per questi due canali dovrebbe essere usato il comando WRITE al posto di PRINT.

```
PRINT "ciao"
ciao                    - finestra 0

PRINT #0,"ciao"
ciao                    - sempre nella finestra 0

PRINT #4,"ciao"
ciao                    - finestra 4 (in cima allo schermo)

PRINT #8,"ciao"
ciao                    - su un'eventuale stampante
```

TAB e SPC sono comodi per formattazioni di stampa semplici; ma per formati più complicati può essere utilizzato il comando PRINT USING con una maschera del formato di stampa che si intende utilizzare. Una maschera è una stringa contenente caratteri speciali ognuno dei quali indica un particolare formato. Questi caratteri, chiamati "specificatori del formato del campo" si possono trovare nella descrizione dettagliata del comando PRINT USING, già presentato. I seguenti esempi dovrebbero però chiarirne il funzionamento.

Ecco i formati disponibili per la stampa delle stringhe:

```
PRINT USING "\    \";"stringa di prova"
string
```

"!" stampa il primo carattere della stringa.

```
PRINT USING "!";"stringa di prova"
s
```

... ma probabilmente il carattere più utile è "&". Questo carattere può essere usato per evitare la caratteristica di a capo automatico del BASIC. Normalmente il BASIC stampa una stringa a capo se non vi è sufficiente spazio per contenerla nella linea corrente. PRINT USING "&" evita questo comportamento.

Si usi BORDER 0 per evidenziare i bordi della finestra.

```
MODE 1:LOCATE 39,1:PRINT "troppo lungo"
                                                            linea 1
troppo lungo                                              - linea 2
```

```
MODE 1:LOCATE 39,1:PRINT USING "&";"troppo lungo"

                                       tr                 - linea 1
oppo lungo                                                - linea 2
```

Per la stampa di numeri sono disponibili molte maschere. Probabilmente la più semplice è PRINT USING "#####" in cui viene stampata una cifra per ogni "#" che appare nella maschera.

```
PRINT USING "######";123
   123
```

Può essere incluso il punto decimale (all'inglese) usando ".".

```
PRINT USING "####.#####";12.45
  12.45000
```

Le cifre prima del punto decimale possono essere raggruppate tre a tre separate da una virgola se nella maschera viene inclusa una virgola (all'inglese) prima del punto decimale.

```
PRINT USING "########,.####";123456.78
  123,456.7800
```

E' inoltre possibile inserire un simbolo di Dollaro o di Lira Sterlina che verrà stampato davanti alla prima cifra del numero, anche se questo non riempie tutta la maschera. Per fare ciò si usano i simboli "$$" e "££".

```
PRINT USING "$$##",7
 $7

PRINT USING "$$##";351
$351

PRINT USING "££####,.##";1234.567
 £1,234.57
```

Si noti che il risultato è stato arrotondato.

Lo spazio prima del risultato può essere riempito da asterischi usando "**" nella maschera.

```
PRINT USING "**####.#";12.22
****12.2
```

Anche i questo caso è possibile inserire un simbolo monetario (anche nella maschera sarà sufficiente specificare un solo simbolo) come in "**$..." e "**£...".

Un "+" all'inizio della maschera fa stampare sempre il segno prima del numero. Un "+" dopo la maschera fa stampare un segno finale.

Il "-" può essere posto solo alla fine della maschera e fa stampare un segno meno finale se il numero è negativo.

```
PRINT USING "+##";12
+12

PRINT USING "+##";-12
-12

PRINT USING "##+";12
12+

PRINT USING "##-";-12
12-

PRINT USING "##-";12
12
```

Un “↑↑↑↑” stampa un numero in forma esponenziale.

```
PRINT USING "###.##↑↑↑↑";123.45
 12.35E+01
```

Usando le maschere per i numeri, si noti che se il numero è troppo lungo, prima del risultato verrà stampato un % mentre il risultato non viene abbreviato per stare nello spazio specificato.

```
PRINT USING "####";123456
%123456
```

Parliamo delle finestre.

Il BASIC del 464/6128 permette di utilizzare fino a otto finestre per il testo. Ogni comando per i testi può essere quindi inviato ad una qualunque di queste finestre.

Il comando usato per posizionare le finestre è WINDOW. Questo è seguito da 5 parametri. Il primo è opzionale e indica la finestra che deve essere definita (se viene omesso, si assume la finestra 0). Il numero della finestra è preceduto dal simbolo di cancelletto. Gli altri 4 numeri indicano i limiti sinistro, destro, superiore ed inferiore della finestra. Questi valori corrispondono a righe e colonne. Le colonne possono andare da 1 a 80, le righe da 1 a 25.

L'esempio seguente definisce la finestra 4 in modo che inizi alla colonna 7 e finisca alla colonna 31, inizi alla riga 6 e finisca alla riga 18. Si reinizializzi il computer e si scriva:

```
WINDOW #4,7,31,6,18
```

Dopo questo comando nulla cambierà sullo schermo; ma si immettano le linee seguenti:

```
INK 3,9
PAPER #4,3
CLS #4
```

Questo farà sì che appaia sullo schermo un largo rettangolo verde, questa è la finestra numero 4. Il comando precedente evidenzia anche il fatto che sia PAPER che CLS possono essere usati con una qualunque delle otto finestre includendo il numero di canale; se questo viene omeso il comando gererà con la finestra 0, quella cioè per difetto.

Ognuno dei seguenti comandi può contenere un indicatore di canale che indica la finestra su cui deve operare:

CLS
COPYCHR$
INPUT
LINE INPUT
LIST
LOCATE
PAPER
PEN
POS
PRINT
TAG
TAGOFF
VPOS
WINDOW
WRITE

La finestra verde che si trova in mezzo allo schermo avrà cancellato parte del testo precedente (posto sulla finestra 0).

E' possibile inviare un testo a qualunque finestra, indicando un canale in un comando PRINT:

PRINT #4,"salve gente"

Queste parole appariranno nell'angolo in alto a sinistra della finestra verde, diversamente dalle parole stampate da questo comando:

PRINT "salve gente"

Se si è inserito quest'ultimo comando, si noterà che parte della finestra verde è stata coperta dal testo.

Se si vuole che i messaggi del BASIC appaiano sulla finestra 4, si può utilizzare il comando WINDOW SWAP:

WINDOW SWAP 0,4

Il "Ready" che seguirà questo comando sarà stampato sulla finestra verde ed il cursore si troverà proprio sotto la scritta. Si provi ora a scrivere:

PRINT #4,"ciao gente"

Le parole "ciao gente" appariranno nella vecchia finestra 0 che è stata scambiata con la finestra 4. Anche le informazioni relative alla posizione del cursore nelle finestre sono state scambiate, perciò anche dopo un WINDOW SWAP i messaggi verranno stampati sotto gli ultimi invece che all'inizio della finestra. Si provi:

```
LOCATE #4,20,1
PRINT "questa è la finestra 0"
PRINT #4,"questa è la finestra 4"
```

Il messaggio "finestra 0" si troverà sulla linea dopo il PRINT; il messaggio "finestra 4" si troverà nella parte centrale della linea superiore dello schermo.

Prima di eseguire un comando WINDOW tutte le otto finestre coprono l'intero schermo. Ciò accade anche dopo l'esecuzione del comando MODE; se dunque il cursore si viene a trovare in una finestra molto piccola è sufficiente scrivere MODE 1 come nell'esempio seguente:

```
MODE 1
WINDOW 20,21,7,18
MO
DE
 1
```

Non ci si preoccupi anche se la parola MODE è spezzata; se si sarà lasciato il necessario spazio tra MODE e 1, il comando funzionerà.

Ora che si conosce qualcosa di più sull'utilizzo delle finestre, si immetta il seguente programma:

```
10 MODE 0
20 FOR n=0 TO 7
30 WINDOW #n,n+1,n+6,n+1,n+6
40 PAPER #n,n+4
50 CLS #n
60 FOR c=1 TO 200: NEXT c
70 NEXT n
```

Questo programma fissa 8 finestre che si sovrappongono ognuna di un diverso colore. Al termine del programma, quando apparirà la parola "Ready" si prema **[RETURN]** alcune volte per vedere il movimento dei blocchi sullo schermo, nonostante questo, le effettive posizioni delle finestre non cambiano. Si provi ora:

```
CLS #4
```

... e si noterà che la finestra 4 è sempre nella stessa posizione. Si osservi ora cosa accade eseguendo:

```
LIST
LIST #4
LIST #3
```

Un'altra caratteristica del comando WINDOW, esemplificata in un programma alla fine del capitolo è che non importa se si specificano i margini destro e sinistro in ordine inverso. Se il valore del primo parametro è maggiore del secondo, il BASIC scambierà i due valori. Lo stesso vale per i bordi superiore ed inferiore.

```
10 MODE 0
20 a=1+RND*19:b=1+RND*19
30 c=1+RND*24:d=1+RND*24
40 e=RND*15
50 WINDOW a,b,c,d
60 PAPER e:CLS
70 GOTO 20
```

Se posso interrompere ...

Forse si sarà già notato che una delle innovazioni più importanti nel software del 464/6128 è costituito dalla possibilità di gestire gli interrupt da BASIC; questo significa che il BASIC riesce ad eseguire simultaneamente ed indipendentemente un certo numero di operazioni all'interno di un programma. Tale possibilità è nota con il nome di multi-tasking ed è implementata dai due comandi AFTER ed EVERY.

Questa possibilità è anche chiaramente dimostrata dal modo in cui è possibile gestire il suono mediante code e rendez-vous.

Ogni aspetto della sincronizzazione si riferisce all'orologio principale del sistema che è un sistema di sincronizzazione al quarzo contenuto nel computer principale che si occupa del tempo e della sincronizzazione degli eventi che avvengono nel computer - operazioni come la scansione dello schermo ed il clock del processore. Quando una funzione dell'hardware è connessa al tempo, può essere seguita utilizzando l'orologio principale al quarzo.

L'implementazione software è costituita dai comandi AFTER e EVERY che, secondo l'approccio rivolto verso l'utente del BASIC AMSTRAD, fanno esattamente ciò che dicono: ad esempio AFTER (DOPO) il momento indicato nella linea di comando, il programma deve eseguire una determinata sub-routine ed eseguire le operazioni indicate.

Il 464/6128 utilizza un orologio in tempo reale. Il comando AFTER permette di eseguire da BASIC delle sub-routine in un dato momento del futuro. Sono disponibili quattro timer di ritardo ognuno dei quali ha associata una sub-routine.

Quando il tempo specificato è trascorso, la sub-routine viene automaticamente chiamata, proprio come se fosse stato eseguito un GOSUB nella posizione corrente del programma. Quando la sub-routine termina, mediante un normale comando RETURN, il programma continua la propria esecuzione da dove era stato interrotto.

Il comando EVERY permette di chiamare delle sub-routine da BASIC ad intervalli regolari. Sono disponibili quattro timer di ritardo ognuno dei quali ha associata una sub-routine.

I timer hanno diverse priorità di interruzione. Il timer 3 ha la proprietà più alta ed il timer 0 quella più bassa.

```
10 MODE 1:n=14:x=RND*400
20 AFTER x,3:GOSUB 80
30 EVERY 25,2 GOSUB 160
40 EVERY 10,1 GOSUB 170
50 PRINT "Controlla i tuoi riflessi"
60 PRINT "premi la barra spaziatrice";
70 IF flag=1 THEN END ELSE 70
80 z=REMAIN(2)
90 IF INKEY$(47)=-1 THEN 110
100 SOUND 1,900:PRINT "baro!":GOTO 150
110 SOUND 129,20:PRINT "ORA!":t=TIME
120 IF INKEY$(47)=-1 THEN 120
130 PRINT "ci hai messo";
140 PRINT (TIME-t)/300;"secondi"
150 CLEAR INPUT:flag=1:RETURN
160 SOUND 1,0,50:PRINT ".";:RETURN
170 n=n+1:IF n>26 THEN n=14
180 INK 1,n:RETURN
```

I comandi AFTER e EVERY possono essere eseguiti in qualunque momento, reinizializzando la routine ed il tempo di ritardo associato ad un dato timer. I timer di ritardo sono gli stessi sia per il comando AFTER che per il comando EVERY, e quindi un AFTER sostituisce ogni EVERY che utilizza un determinato timer e viceversa.

I comandi DI ed EI dibabilitano o abilitano i timer e gli interrupt per il suono mentre i comandi all'interno di essi vengono eseguiti. Questo ha l'effetto di ritardare l'interrupt di proprietà più alta del timer 1 da tutto ciò che potrebbe accadere durante la gestione dell'interrupt da parte del timer con priorità più bassa.La funzione REMAIN disabilita e restituisce il tempo mancante per uno dei quattro timer.

Uso dei DATA ...

In un programma che usa sempre lo stesso gruppo di informazioni che devono essere inserite all'inizio, potrebbe essere utile usare un meccanismo che eviti di chiedere ogni volta all'utente le stesse informazioni. Questa possibilità è offerta dai comandi READ e DATA. READ, che significa LEGGI, assegna valori alle variabili. Diversamente da INPUT prende valori dalle linee DATA. I due esempi seguenti ne mostrano il funzionamento.

```
10 INPUT "inserire 3 numeri separati da virgole";a,b,c
20 PRINT "i numeri sono";a;"e";b;"e";c
run
```

```
10 READ a,b,c
20 PRINT "i numeri sono";a;"e";b;"e";c
30 DATA 12,14,21
run
```

Anche nelle linee DATA i diversi elementi sono separati da virgole.

In una linea DATA è possibile inserire anche stringhe:

```
10 DIM a$(11)
20 FOR i=0 TO 11
30 READ a$(i)
40 NEXT
50 FOR i=0 TO 11
60 PRINT a$ (i) ;" ";
70 NEXT
80 DATA tanto,va,la,gatta,al,lardo,che,ci,lascia,lo,zampino
run
```

Anche se la linea DATA contiene stringhe, queste non devono essere racchiuse tra doppi apici ""; in effetti in questo caso il loro uso è opzionale come pure lo è inserendo una stringa in risposta ad un INPUT. E' utile usare i doppi apici nel caso in cui una stringa contiene una virgola. Infatti, se non vi fossero i doppi apici, la virgola verrebbe interpretata come un delimitatore.

```
10 READ a$
20 WHILE a$<>"*"
30 PRINT a$
40 READ a$
50 WEND
60 DATA La vecchia, desolata, malconcia casa scricchiolava al vento
70 DATA "L'uomo alto, magro, scuro tossı' violentemente"
80 DATA *
run
```

La stringa della linea 60 contiene delle virgole e dunque ogni parte verrà letta e stampata separatamente. La stringa alla linea 70 è delimitata dai doppi apici e verrà stampata per intero.

L'esempio qui sopra mostra che le linee di dati possono trovarsi su più linee. READ leggerà le linee in ordine (60,70,80,ecc). Un altro fatto non altrettanto ovvio è che le linee DATA possono trovarsi in qualunque punto del programma, prima o dopo il READ che le legge.

Se un programma contiene più READ, il secondo inizierà a leggere dove il primo ha finito:

```
10DATA 123,456,789,321,654,2343
20 FOR i=1 TO 5
30 READ num
40 totale=totale+num
50 NEXT
60 READ totale2
70 IF totale=totale2 THEN PRINT "i dati sono ok" ELSE PRINT "errore nei dati"
run
```

Si provi ad editare la linea 10, si modifichi uno dei primi 5 numeri e si esegua nuovamente il programma. Questa tecnica consiste nell'aggiungere al termine della riga, un numero che sia la somma degli altri. In questo modo è facile trovare eventuali errori, specie se vi sono molte linee DATA.

Se in un programma vi sono DATA misti (stringhe e numeri) non vi è alcun problema; occorre però che i dati vengano letti correttamente secondo il tipo. Ad esempio, se una linea DATA contiene nell'ordine due numeri ed una stringa, è necessario leggere prima i due numeri e poi la stringa:

```
10 DIM a(5),b(5),s$(5)
20 FOR i=1 TO 5
30 READ a(i),b(i),s$(i)
40 NEXT
50 DATA 1,7,marco,3,9,davide,2,2,luca,4,6,paolo,9,1,alfonso
60 FOR i=1 TO 5
70 PRINT s$(i),":"; a(i)*b(i)
80 NEXT
```

Se invece si vogliono separare i diversi tipi di dati:

```
10 DIM a(5),b(5),s$(5)
20 FOR i=1 TO 5
30 READ a(i),b(i)
40 NEXT
50 FOR i=1 TO 5
60 READ s$(i)
70 NEXT
80 DATA 1,7,3,9,2,2,4,6,9,1
90 DATA marco,davide,luca,paolo,alfonso
100 FOR i=1 TO 5
110 PRINT s$(i),":"; a(i)*b(i)
120 NEXT
```

Se si cambia la linea 20 in:

```
20 FOR i=1 TO 4
```

... i primi due tentativi di leggere una stringa alla linea 60 leggeranno "9" e "1". Anche se questi valori possono essere considerati delle stringhe, il risultato non è quello che ci si doveva attendere. Per fare in modo che il programma funzioni correttamente, si possono includere i due comandi seguenti:

```
15 RESTORE 80
45 RESTORE 90
```

Il comando RESTORE porta il puntatore che indica il dato che deve essere letto da un READ alla linea specificata. E' possibile inserirlo in un'istruzione condizionale per leggere un certo blocco di DATA a seconda del verificarsi di certe condizioni. Ad esempio, in un gioco a più livelli che si sviluppa su più schermi, i DATA relativi ad ogni schermo possono essere prelevati considerando il valore di una certa variabile, ad esempio "livello". Segue un esempio di questo tipo.

```
1000 REM disegno dello schermo
1010 IF livello=1 THEN RESTORE 2010
1020 IF livello=2 THEN RESTORE 2510
1030 IF livello=3 THEN RESTORE 3010
1040 FOR y=1 TO 25
1050 FOR x=1 to 40
1060 READ carattere
1070 LOCATE x,y:PRINT CHR$(carattere);
1080 NEXT x,y
:
2000 REM Dati schermo 1
2010 DATA 200,190,244,244,210,...ecc
:
2500 REM Dati schermo 2
2510 DATA 100,103,245,243,251,...ecc
:
3000 REM Dati schermo 3
3010 DATA 190,191,192,193,194,...ecc
```

Ecco un altro esempio di uso di DATA, READ e RESTORE per suonare un motivo. I periodi vengono letti da istruzioni READ all'interno di linee DATA; RESTORE fa ripetere il motivo.

```
10 FOR i=1 TO 3
20 RESTORE 100
30 READ note
40 WHILE nota<>-1
50 SOUND 1,nota,35
60 READ nota
70 WEND
80 NEXT
90 SOUND 1,142,100
100 DATA 95,95,142,127,119,106
110 DATA 95,95,119,95,95,119,95
120 DATA 95,142,119,142,179,119
130 DATA 142,142,106,119,127,-1
run
```

Il sound della musica ...

Di tutte le possibilità offerte dal 464/6128, i vari comandi relativi al suono e agli inviluppi potranno sembrare i più ostici, i più difficili da ricordare ma non è necessariamente così. Con un po' di pratica sarà possibile ottenere vari rumori e perfino far suonare alla macchina un brano completo.

Guardiamo innanzitutto le prime 4 parti del comando SOUND. Queste sono: il numero del canale, il periodo del tono,la durata della nota e il volume. Ora occorre sapere che valori possono assumere tali parametri.

Lasciamo per il momento perdere il primo parametro (numero del canale) che è piuttosto complicato. La seconda parte (periodo del tono) può essere un qualunque valore intero compreso tra 0 e 4095, ma solo alcuni dei valori compresi tra questi estremi producono una nota musicale riconoscibile. Si veda la Parte 5 del Capitolo "Riferimenti". Ad esempio il numero 239 suonerà il Do centrale mentre il numero 253 suona un Si appena al di sotto del Do centrale. I valori da 240 a 252 eseguono una scala che però non corrisponde ad una scala del piano. Se il periodo del tono è 0, non viene emesso alcun suono - tornerà utile quando si parlerà dei rumori.

La terza parte del comando SOUND fornisce la durata della nota in centesimi di secondo. Questo valore può essere compreso tra 1 e 32767. Tuttavia, se il valore è 0, la lunghezza della nota verrà determinata dall'inviluppo usato (si vedrà questo più avanti). Se il valore è negativo, indica quante volte l'inviluppo deve essere suonato (perciò -3 indica di ripetere l'inviluppo del volume 3 volte, anche questo verrà spiegato più avanti).

La quarta parte del comando è il volume. Questo può andare da 0 a 15 (se viene omesso si assumerà 12). In tutti gli esempi che si sono visti finora, il volume rimaneva costante per tutta la durata della nota. Usando un inviluppo del volume, la parte relativa al volume del comando SOUND viene considerata il valore iniziale della nota.

Parliamo ora del numero del canale. Questo è un numero che assume significato bit a bit e sarà dunque necessario conoscere qualcosa sui numeri binari per comprendere appieno l'argomento; si veda dunque la parte 1 di questo Capitolo.

Un suono può essere emesso da uno qualunque dei tre generatori; se il computer è collegato ad un amplificatore stereo, un canale si troverà a sinistra, uno a destra ed il terzo in centro (sia a destra che a sinistra). Per scegliere il canale che dovrà emettere la nota si usano i seguenti tre numeri:

1 canale A
2 canale B
3 canale C

Per suonare con più di un canale, occorre sommare i numeri dei canali desiderati. Ad esempio, per suonare sui canali A e C si usa 1+4=5.

```
SOUND 5,284
```

Potrebbe sembrare strano che al canale C sia stato dato il numero 4 invece del numero 3. Ciò avviene perchè ognuno di questi numeri è una potenza di 2 (1=2^0, 2=2^1, 4=2^2) e perciò è possibile combinarli per formare un numero binario. Se si pensa a tre numeri binari, ognuno dei tre numeri può assumere i valori 0 o 1 ed in questo modo si indica se un dato canale deve essere attivo o no. Dall'esempio appena visto:

5 in decimale è l'equivalente di 1*4+0*2+1*1 o 101 in binario. Se ad ogni cifra del numero binario associamo un canale, avremo:

C B A
1 0 1

In altri termini, il canale C è attivo, il B no e l'A è attivo. Se si pensa di emettere la nota dai canali A e B, bisogna quindi utilizzare il numero

C B A
0 1 1

Il numero binario 011 corrisponde a 0*4+1*2+1*1=3. Il comando avrebbe perciò dovuto essere:

```
SOUND 3,142
```

Questo è naturalmente lo stesso numero che si sarebbe ottenuto sommando i valori dei canali interessati (si ricordi A=1, B=2, C=4). Per suonare con i canali A e B, il numero da utilizzare è dunque 1+2=3.

Se non si è compreso a fondo questo argomento, non ci si preocupi. E' sufficiente sapere che per far suonare più canali bisogna sommare i valori corrispondenti a ciascun canale.

Sfortunatamente vi sono anche altri valori che entrano a far parte del numero di canale. I numeri 8, 16 e 32 indicano che il suono deve effettuare un rendezvous con un altro canale (rispettivamente A, B e C). Ci si starà probabilmente chiedendo in che senso si parla di rendezvous. Fino ad ora i suoni prodotti andavano direttamente ad un dato canale. Si provi ora:

```
SOUND 1,142,2000
SOUND 1,90,200
```

Se non si sarà stati molto lenti ad immettere il secondo comando, ci si sarà accorti che si è potuto inserire quest'ultimo prima che il precedente finisse di suonare. Ciò avviene perchè il sistema di generazione sonora può memorizzare fino a 5 comandi sonori in una coda. Se desideriamo emettere un suono dal canale A e poi due suoni simultaneamente dai canali A e B, occorrerà trovare un modo per far conoscere al computer che la seconda nota non dovrebbe iniziare prima che sia finito il primo suono del canale A; un canale dovrà attendere l'altro. Questa operazione è nota con il nome di *rendezvous* e vi sono due modi per utilizzarlo:

```
SOUND 1,200,1000
SOUND 3,90,200
```

In questo caso la seconda nota è diretta ad A e B e dunque non può iniziare prima che finisca la prima nota sul canale A. La limitazione imposta da questo comportamento (una combinazione di note deve attendere che tutti i canali su cui opera siano liberi) è che ad entrambi i canali viene inviato lo stesso suono (in questo caso ,90,200 è stato suonato sia da A che da B). Il metodo alternativo consiste in:

```
SOUND 1,200,2000
SOUND 1+16,90,200
SOUND 2+8,140,400
```

In questo caso la seconda nota sul canale A è posta in rendezvous con il canale B ed il canale B è posto in rendezvous con il canale A. Il vantaggio è ovvio: anche se la seconda nota emessa da A è diversa dalla nota emessa da B, esse sono collegate tra di loro in modo che non possano iniziare se entrambi i canali non sono liberi - questo è il *rendezvous*. Anche in questo caso il valore deve essere letto bit a bit:

8=2^3, 16=2^4, 32=2^5

... il numero di canale può dunque essere visto come un numero binario in cui le cifre hanno i seguenti significati:

rendezvous	rendezvous	rendezvous	suono	suono	suono
C	B	A	C	B	A
+32	+16	+8	+4	+2	+1

Per far emettere una nota da C in rendezvous con A, si usa dunque:

0	0	1	1	0	0

.. in binario 1100, uguale a 8+4=12

Perciò un numero di canale uguale a 12 chiede al computer di emettere una nota dal canale C ed attendere una nota con cui vi è un rendezvous sul canale A.

Se si aggiunge al numero di canale il numero 64 (2↑6), significa che la nota deve essere trattenuta. In pratica significa che la nota non deve essere emessa fino all'esecuzione del comando RELEASE.

Infine, utilizzando 128 (2↑7) la coda del canale specificato viene gettata via. Perciò, se si inizia un suono che sta durando troppo su un determinato canale, un modo rapido per fermarlo è:

```
SOUND 1,248,30000        (è un suono che dura 5 minuti)
SOUND 1+128,0            (pone termine al suono)
```

In modo diretto, vi è un modo più rapido di terminare un lungo suono: consiste nel premere all'inizio della linea il tasto **[DEL];** il breve bip elimina tutte le code dei canali.

Ora che possiamo inviare un qualunque suono ad uno qualunque dei tre canali (con eventuali rendezvous), sarebbe bello poter produrre un suono più melodioso del puro e semplice bip. Per fare ciò si utilizzano gli inviluppi che definiscono il modo in cui la nota deve suonare più o meno forte nella sua durata. Una nota prodotta da uno strumento ha un momento di attacco, in cui il volume cresce molto rapidamente. Il volume scende poi ad un determinato livello che viene mantenuto per un certo tempo, passato il quale il suono cala lentamente verso lo zero. Alle note prodotte con SOUND è possibile fornire un inviluppo di questo tipo. Per fare ciò si usa la parola chiave ENV. Vediamo un semplice esempio:

```
ENV 1,5,3,4,5,-3,8
SOUND 1,142,0,0,1
```

Il comando ENV deve venire prima del comando SOUND su cui opera. Per usare questo inviluppo in un comando SOUND, il numero dell'inviluppo deve essere aggiunto come quinto parametro - in questo caso 1). Il primo numero di un comando ENV è il numero dell'inviluppo che si intende definire. L'istruzione ENV indica poi la durata della nota e le sue caratteristiche di volume, e dunque i parametri relativi alla durata ed al volume del comando SOUND sono a zero. L'inviluppo definito or ora fa crescere il suono in 5 passi in ognuno dei quali il suono aumenta di 3 unità ed in cui ogni passo dura 4 centesimi di secondo. Il volume viene poi ridotto in 5 passi, in ognuno dei quali il volume cala di 3 unità ed in cui ogni passo dura 8 centesimi di secondo. In altre parole il primo numero indica l'inviluppo che si intende definire; questo numero è seguito da due gruppi di tre numeri. In ognuno dei due gruppi il primo numero indica in quanti passi il volume deve salire o scendere, il secondo indica di quante unità il volume deve aumentare o calare ad ogni passo ed il terzo indica la durata di ogni passo.

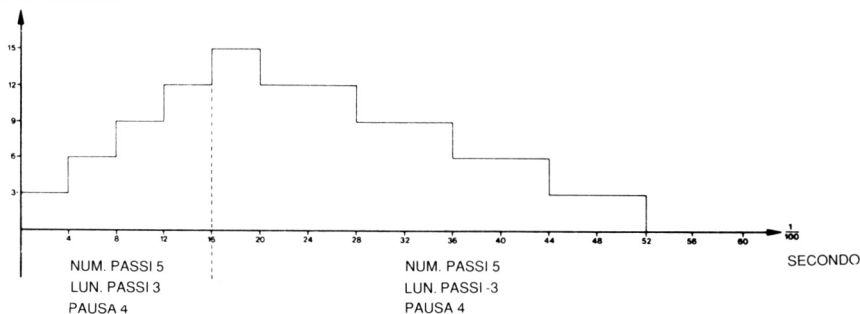
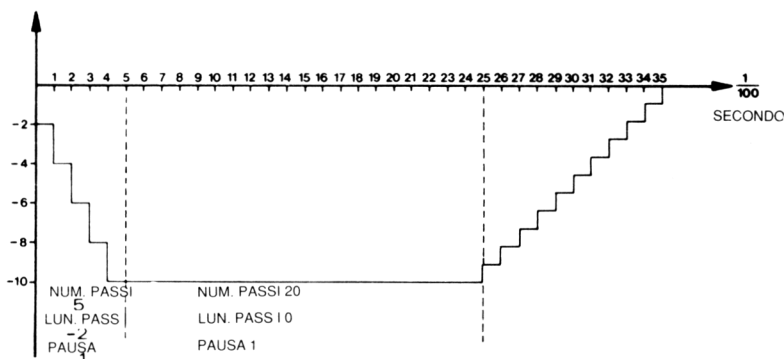
La durata totale di ognuna delle due parti è uguale al primo valore (numero dei passi) moltiplicato per il terzo (durata dei passi). La crescita o decrescita totale del volume è uguale al numero dei passi per le unità di volume che vengono aggiunte o tolte ad ogni passo. La durata media di un inviluppo con più di una parte è uguale alla somma delle durate delle sue parti.

Naturalmente il volume iniziale non deve necessariamente essere uguale a 0 (come nel comando SOUND). L'esempio precedente presentava una nota in cui il volume prima saliva e poi scendeva. L'esempio seguente presenta una nota che prima cala e poi cresce:

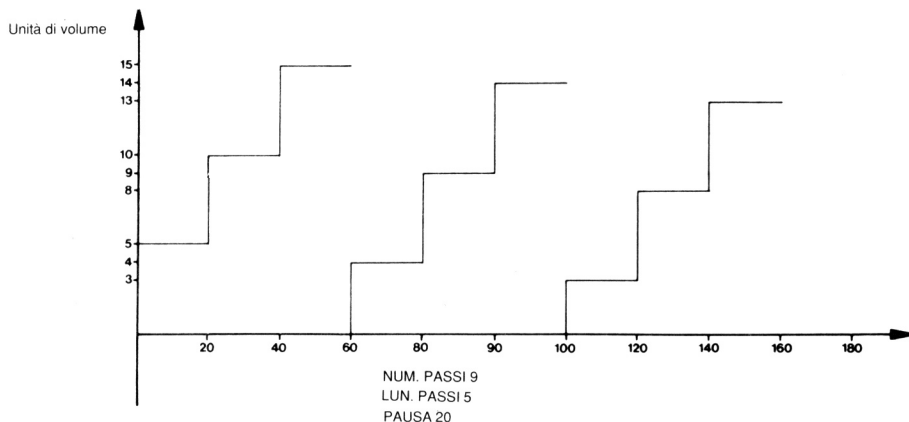
ENV 2,5,-2,1,20,0,1,10,1,1
SOUND 1,248,0,15,2

Questo inviluppo diventa il numero 2 ed è formato da tre parti. Nella prima il volume viene ridotto di due unità ogni 5 passi della durata di 1 centesimo di secondo. La seconda parte consiste di 20 passi di un centesimo di secondo ma con nessun incremento o riduzione del volume (volume costante). La terza parte è formata da 10 passi di 1 centesimo di secondo l'uno in cui il suono aumenta di 1 unità per passo.

Il comando SOUND ha un volume iniziale di 15 e dunque dopo la prima parte verrà ridotto a 5 dove rimane costante per 20 centesimi di secondo ed alla fine viene riportato a 15 nella parte finale del'inviluppo.

E' piuttosto difficile visualizzare la forma di questo inviluppo ma può aiutare disegnarlo prima su carta e determinare quindi i parametri che devono essere passati al comando ENV. Le figure successive mostrano la forma dei due inviluppi prodotti finora:

Un inviluppo può essere composto da un massimo di 5 parti, ognuna delle quali è formata da 3 valori, e dunque il comando ENV può contenere fino a 16 parametri (compreso il primo che indica l'inviluppo utilizzato - 1..15). Se il volume dovesse superare il 15 o andare sotto lo 0, il passo successivo al 15 verrà assunto 0 ed il passo sotto lo 0 diventerà il 15:

```
ENV 3,9,5,20
SOUND 1,142,0,0,3
```

Questo semplice inviluppo produce 9 passi della durata di 20 centesimi di secondo in ognuno dei quali il volume aumenta di 5 unità. Dopo i primi 3 passi, il volume sarà uguale a 15 e i passi successivi lo porteranno a 4,9 e così via, Il diagramma seguente ne mostra gli effetti:

Il numero dei passi può andare da 0 a 127. La variazione del volume dei passi può andare da -128 a 127 (i valori negativi fanno calare il volume) e la durata dei passi può andare da 0 a 255.

Ora che possiamo creare un inviluppo del volume che definisce la "forma" del suono, potremmo voler modificare il tono per produrre effetti di "vibrato" in cui la nota varia intorno alla frequenza principale.

Questo controllo viene effettuato da un comando molto simile a quello di controllo del volume. Gli inviluppi dei toni vengono prodotti usando il comando ENT. Ad esempio:

```
ENT 1,5,1,1,5,-1,1
SOUND 1,142,10,15,,1
```

Si utilizza l'inviluppo del tono nel comando SOUND inserendolo come sesto parametro del comando SOUND. Anche in questo caso il comando ENT deve precedere il comando SOUND.

Il primo esempio di ENT specifica che nell'inviluppo di tono numero 1 vi devono essere 5 passi in ognuno dei quali il periodo aumenta di 1; ogni passo dura 1 centesimo di secondo. La seconda parte dell'inviluppo specifica altri 5 passi in cui il periodo del tono cala di 1 unità (-1) ed in cui ogni passo dura 1 centesimo di secondo. La lunghezza totale è dunque di 5+5=10 centesimi di secondo. Si noti che la durata della nota deve essere specificata nel comando SOUND poiché l'inviluppo del tono non specifica la durata della nota (contrariamente all'inviluppo del volume). Se la lunghezza specificata nel comando SOUND è più corta della durata dell'inviluppo del tono, l'ultima parte dell'inviluppo verrà persa. Se è maggiore, l'ultima parte della nota continuerà ad un livello costante. Ciò avviene anche se per determinare la lunghezza della nota si usa l'inviluppo del volume.

Si noti nell'esempio sopra l'assenza di un riferimento ad un inviluppo del volume. Ciò avviene poiché non abbiamo creato un inviluppo del volume per questo suono.

La maggior parte degli inviluppi dei toni sarà probabilmente più corta della lunghezza della nota. Per questo motivo è possibile far ripetere l'inviluppo del tono per tutta la durata della nota. Si può fare ciò specificando un inviluppo negativo il cui corrispondente valore positivo viene usato dal comando SOUND:

```
ENT -5,4,1,1,4,-1,1
SOUND 1,142,100,12,,5
```

Questo inviluppo produrrà un effetto di vibrato nella nota. Quando si definisce un inviluppo per i toni, è conveniente farlo variare simmetricamente rispetto al periodo iniziale del tono in modo che, quando viene ripetuta, la nota non si allontanerà troppo dal suo valore iniziale. Si provi:

```
ENT -6,3,1,1
SOUND 1,142,90,12,,6
```

Si noterà che questo suono cala molto rapidamente in frequenza in quanto ogni volta che l'inviluppo viene ripetuto il periodo cresce di 3 unità e ciò avviene 30 volte (90/3). Questo effetto può tornare utile in suoni come fischi e sirene:

```
ENT -7,20,1,1,20,-1,1
SOUND 1,100,400,12,,7

ENT -8,60,-1,1,60,1,1
SOUND 1,100,480,12,,8
```

E' possibile definire contemporaneamente fino a 15 inviluppi del tono (da 1 a 15); come si è detto un numero negativo indica che l'inviluppo deve essere ripetuto. Il numero di passi (il primo valore in ogni gruppo di 3) può andare da 0 a 239. Come nell'inviluppo del volume, le dimensioni di ogni passo possono andare da -128 a 127 ed il tempo di pausa può andare da 0 a 255. Anche il comando ENT come il comando ENV può contenere un massimo di 5 parti ognuna contenente 3 valori.

L'ultima parte che può essere inserita nel comando SOUND è il settimo parametro che indica il livello del rumore che deve essere sommato al suono. Bisogna ricordare che vi è un solo generatore di rumore e quindi ogni periodo del rumore elimina il precedente.

Aggiungendo un rumore ad una nota le si dà un suono differente; è anche possibile usare il rumore da solo indicando un periodo del tono (secondo parametro del comando SOUND) uguale a 0 che significa la presenza del solo rumore. Ciò può essere utilizzato per suoni percussivi. Si provi:

```
ENT -3,2,1,1,2,-1,1
ENV 9,15,1,1,15,-1,1
FOR a=1 TO 10:SOUND 1,4000,0,0,9,3,15:NEXT
```

Questa potrebbe essere la base del rumore di un treno. Per fare ciò vengono utilizzati entrambi gli inviluppi ed il rumore.

Le parti relative alla durata ed al volume del comando SOUND sono entrambe a 0 in quanto tali valori vengono determinati dall'inviluppo del volume.

Ora che abbiamo imparato ad usare tutti gli aspetti dei comandi SOUND, ENV ed ENT, possiamo esaminare altri comandi e funzioni ad essi associati.

Quando abbiamo descritto il numero di canale nel comando sound, aggiungendo 64 ad esso, il suono veniva trattenuto e quindi rimaneva in coda finché non fosse stato rilasciato. L'operazione di rilascio del suono viene effettuata dal comando RELEASE. Tale parola è seguita da un numero che ha significato bit a bit in cui ogni bit indica quale canale occorre rilasciare. Ancora una volta occorre comprendere che:

4 significa canale C
2 significa canale B
1 significa canale A

... ed è possibile rilasciare più canali sommando tra di loro questi numeri. Per rilasciare dei suoni trattenuti in tutti e tre i canali, si usa:

```
RELEASE 7
```

... dove 7=1+2+4. Se nessun canale è trattenuto, il comando RELEASE viene ignorato. Si provi ora:

```
SOUND 1+64,90
SOUND 2+64,140
SOUND 4+64,215
RELEASE 3:FOR t=1 TO 1000:NEXT:RELEASE 4
```

I suoni che ci si attenderà dai tre comandi SOUND non verranno emessi finché non viene eseguito il primo RELEASE che permette di eseguire i suoni dei canali A e B. Dopo un certo ritardo verrà rilasciato anche il suono del canale C.

Vi è un altro modo in cui più suoni possono incontrarsi in un rendezvous. Quando viene aggiunto alla coda un suono con il bit di trattenimento (con sommato il numero 64), non verrà trattenuto solo quel suono, ma anche tutti quelli che verranno successivamente inviati alla coda. Se vengono inviati altri 4 suoni, la macchina si fermerà finché non verranno rilasciati i suoni, eventualmente con una subroutine chiamata dopo un periodo di tempo fissato (usando AFTER o EVERY). Tuttavia questo non è un buon metodo per usare il sistema sonoro in quanto il programma contenente i comandi SOUND può fermarsi di tanto in tanto nel momento in cui la coda si riempie. Ciò può avvenire anche se vengono accodati in rapida successione molti suoni di lunga durata. Si provi:

```
10 FOR a=1 TO 8
20 SOUND 1,100*a,200
30 NEXT
40 PRINT "ciao"
run
```

Si noterà che la parola "ciao" non appare istantaneamente ma dopo i primi tre suoni. Ciò avviene in quanto il programma non può continuare finché non si libera uno spazio nella coda.

Il BASIC contiene un meccanismo di interruzioni (interrupt) simile a quello usato nei comandi AFTER ed EVERY ed in ON BREAK GOSUB. Ciò permette di specificare una routine relativa al suono che viene richiamata solo quando vi è spazio libero nella coda. Si provi:

```
10 a=0
20 ON SQ(1) GOSUB 1000
30 PRINT a;
40 GOTO 30
1000 a=a+10
1010 SOUND 1,a,200
1020 IF a<200 THEN ON SQ(1) GOSUB 1000
1030 RETURN
run
```

Si noti che il programma non si ferma mai. Il comando SOUND viene chiamato solo quando la coda del canale A (il numero 1) ha uno spazio libero. Questa condizione viene controllata da ON SQ(1) GOSUB alla linea 20. Il comando inizializza un meccanismo di interrupt che richiama la sub-routine di generazione sonora nonappena si libera uno spazio nella coda specificata. Dopo essere stato usato, ON SQ GOSUB deve essere reinizializzato e questa operazione viene effettuata dalla linea 1020 della sub-routine. In questo esempio la sub-routine si reinizializza se il valore di a è minore di 200.

In un programma completo che può spostare oggetti sullo schermo o fare somme e così via, è possibile avere una musica di sottofondo richiamando la sub-routine solo quando vi è uno spazio libero nella coda. In questo modo il programma non si fermerà in attesa che si liberi uno spazio della coda. Se i valori delle none vengono presi da delle linee DATA, si può fare in modo che la sub-routine venga rieseguita nonappena i dati sono finiti.

Il numero all'interno delle parentesi di ON SQ( ) GOSUB può essere 1,2 o 4, a seconda del canale che viene controllato.

Per leggere lo stato di un qualunque canale dall'interno di un programma, si può usare la funzione SQ(). Il numero all'interno delle parentesi può anche in questo caso essere 1,2 o 4 a seconda del canale a cui si è interessati, Questa funzione restituisce un valore che deve essere letto bit a bit e per decifrarlo occorre conoscere i numeri binari. I bit del valore restituito hanno il seguente significato:

BIT	DECIMALE	SIGNIFICATO
0,1,2	1,2,4	Numero di posizioni libere nella coda
3	8	La nota in cima alla coda effettua un rendezvous con A
4	16	La nota in cima alla coda effettua un rendezvous con B
5	32	La nota in cima alla coda effettua un rendezvous con C
6	64	La nota in cima (ed anche la coda) è trattenuta
7	128	Una nota è in esecuzione

Si provi quest'esempio:

```
10 SOUND 2,200
20 x=SQ(2)
30 PRINT BIN$(x)
run
```

Verrà stampato il numero binario 10000100, in cui il bit 7 è uguale a 1 (il canale sta emettendo una nota al momento dell'esecuzione di SQ( )). Le ultime tre cifre (100) convertite in decimale danno il valore 4 e quindi vi sono 4 posti liberi nella coda. Questa funzione può essere utilizzata per controllare lo stato di una coda in un determinato punto del programma diversamente da ON SQ( ) GOSUB che effettua un controllo e risponderà allo stato della coda in un punto indeterminato del programma.

Fino ad ora tutti gli esempi riguardavano solo una o due note. E' possibile emettere un intero gruppo di note distinte per ottenere un brano musicale, elencando le note in istruzioni DATA; queste verranno poi lette da istruzioni READ e suonate da istruzioni SOUND.

```
10 FOR ottava=-1 TO 2
20 FOR x=1 TO 7:REM 7 note per ottava
30 READ nota
40 SOUND 1,nota/2↑ottava
50 NEXT
60 RESTORE
70 NEXT
80 DATA 426,379,358,319,284,253,239
run
```

L'ultimo esempio sviluppa ulteriormente questo principio. Le note ed i ritmi vengono suonati dai canali A e B, usando i rendezvous per sincronizzarsi. L'esempio dimostra uno dei modi in cui è possibile formattare istruzioni DATA per includere note,ottave,lunghezze ed informazioni sui redezvous:

```
10 REM la linea 180 contiene i suoni alti
20 REM la linea 190 contiene i suoni bassi
30 DIM scala%(12):FOR x%=1 TO 12: READ scala%(x%):NEXT
40 can1%=1:READ can1$:can2%=1:READ can2$
50 CLS
60 spd%=12
70 scala$=" a-b b c+c d-e e f+f g+g"
80 ENV 1,2,5,2,8,-1,10,10,0,15
90 ENV 2,2,7,2,12,-1,10,10,0,15
100 ENT -1,1,1,1,2,-1,1,1,1,1
110 DEF FNM$(s$,s)=MID$(s$,s,1)
120 can1%=1:GOSUB 200
130 can2%=1:GOSUB 380
140 IF can1%+can2%>0 THEN 140
150 END
160 DATA &777,&70c,&6a7,&647,&5ed,&598
170 DATA &547,&4fc,&4b4,&470,&431,&3f4
180DATA   4cr4f4f1f1g1A1-B2C2f4g2g1A1-B6A2Cr1f1g1f1g1a1-b1A1-
          b2C2g2A2g2f1g1a2g2f6e2c2e2c2
          g2e2c1-B1A2g2f4e4d8c4f3f1c2d4-b2fr2-B2A2g2f6e2gr4C4-
          B1a1f1-b1g2c2-b4a4g4fr6A2A2-B4-B2Ar2-B2A2g2f6e2g4C4-
          B1A1f1-B1g2C2-B4A4g8f.
190DATA   r4f4f8f4e4c4fr8f4e2f2e4d2e2d8c8c6e2f4g4g8e4f3f
          1c4dr8g4cr4e4c6f2d4c4c8fr8-e4dr8g
          8c4e4c6f2d4c4c8f.
200 REM invia il suono al canale A
210 p1$=FNM$(can1$,can1%)
220 IF p1$<>"r" THEN r1%=0:GOTO 240
230 r1%=16:can1%=can1%+1:p1$=FNM$(can1$,can1%)
```

continua nella prossima pagina

```
240 IF p1$="." THEN can1%=0:RETURN ELSE l1%=VAL(p1$)
250 can1%=can1%+1
260 n1$=FNM(can1$,can1%)
270 can1%=can1%+1
280 IF n1$="+" OR n1$="-" THEN 350
290 n1$=" "+n1$
300 nd1%=(1+INSTR(scala$,LOWER$(n1$)))/2
310 IF ASC(RIGHT$(n1$,1))>96 THEN o1%=8 ELSE o1%=16
320 SOUND 1+r1%,scala%(nd1%)/o1%,spd%(l1%,0,1,1
330 ON SQ(1) GOSUB 200
340 RETURN
350 n1$=n1$+FNM(can1$,can1%)
360 can1%=can1%+1
370 GOTO 300
380 REM invia il suono al canale B
390 p2$=FNM$(can2$,can2%)
400 IF p2$<>"r" THEN r2%=0:GOTO 420
410 r2%=8:can2%=can2%+1:p2$=FNM$(can2$,can2%)
420 IF p2$="." THEN can2%=0:RETURN ELSE l2%=VAL(p2$)
430 can2%=can2%+1
440 n2$=FNM(can2$,can2%)
450 can2%=can2%+1
460 IF n2$="+" OR n2$="-" THEN 530
470 n2$=" "+n2$
480 nd2%=(1+INSTR(scala$,LOWER$(n2$)))/2
490 IF ASC(RIGHT$(n2$,1))>96 THEN o2%=4 ELSE o2%=8
500 SOUND 2+r2%,scala%(nd2%)/o2%,spd%*l2%,0,2
510 ON SQ(2) GOSUB 380
520 RETURN
530 n2$=n2$+FNM(can2$,can2%)
540 can2%=can2%+1
550 GOTO 480
run
```

Parliamo di grafica

Questa parte descrive le possibilità grafiche del 464/6128. Il primo esempio verrà via via ampliato per mostrare le varie operazioni possibili.

Per iniziare divideremo lo schermo in una finestra di testo (linea 40) e una finestra grafica (linea 30), fissando il MODE ed una coppia di colori lampeggianti (linea 20):

```
10 REM programma grafico
20 MODE 1:INK 2,10,4:INK 3,4,10
30 ORIGIN 440,100,440,640,100,300
40 WINDOW 1,26,1,25
50 CLG 2
```

Eseguendo questo programma apparirà un quadrato lampeggiante nella parte centrale del lato destro dello schermo. Questo quadrato ha colore 2 (magenta azzurro lampeggianti) fornitogli alla linea 50 e l'origine delle coordinate è stata portata nell'angolo inferiore sinistro del quadrato. Il comando MODE ha portato il cursore grafico all'origine delle coordinate (X=0,Y=0) ed in tal modo possiamo disegnare una riga diagonale sullo schermo con la linea 60:

```
60 DRAW 200,200,3
```

Si esegua il programma modificato e si veda il risultato. Ora si aggiunga:

```
80 MOVE 0,2:FILL 3
```

La linea 80 porta il cursore grafico all'interno del quadrato e lo riempie con il colore 3. I limiti del riempimento coincidono con i bordi della finestra grafica (in questo caso coincidono con i bordi del quadrato) e tutto ciò che è stato disegnato con il colore della penna (3) o con l'inchiostro usato da FILL (anch'esso 3).

Ora si esegua nuovamente il programma.

Per vedere più chiaramente il comportamento di FILL ai bordi, si aggiunga la linea 70. Si noti che è solo il fatto che lo riempimento viene effettuato con lo stesso colore della linea diagonale che restringe il riempimento a metà del quadrato.

```
70 GRAPHICS PEN 1
run
```

Si editi ora la linea 80 per effettuare un FILL con il colore 1 poi si esegua nuovamente il programma per dimostrare quest'ultimo punto. Si ritorni poi a FILL 3.

Ora si aggiungano le linee da 100 a 140 che disegnano un quadrato:

```
100 MOVE 20,20
110 DRAW 180,20
120 DRAW 180,180
130 DRAW 20,180
140 DRAW 20,20
run
```

Il riquadro viene disegnato con il colore 1 a causa della linea 70. Eliminando la linea 70 avremmo dovuto aggiungere ",1" come terzo parametro del comando MOVE della linea 100 o come terzo parametro della linea 110 per chiedere al computer di cambiare penna.

Uniamo i punti ...

Le linee non devono necessariamente essere continue, possono essere tratteggiate. Il comando MASK permette di specificare le dimensioni dei punti. La maschera verrà ripetuta ogni 8 pixel ed ogni linea successiva utilizzerà lo stesso schema. Un nuovo comando MASK (con lo stesso parametro di quello attualmente in uso) farà ritornare ai normali 8 pixel.

La maschera di punti è in effetti costituita da un numero binario di un solo byte dove i bit a 1 indicano i punti in cui utilizzare la penna. Nel nostro esempio useremo una costante binaria (indicata da &X) che chiede di colorare 4 pixel al centro degli 8 pixel con i due più esterni non colorati. Perciò si aggiunga:

```
90 MASK &X00111100
run
```

Ma un momento! Questo programma negli angoli non funziona come ci si aspetterebbe. Il motivo risiede nel fatto che i punti negli angoli vengono in effetti stampati due volte: una come ultimo pixel di una linea ed un'altra come primo pixel della linea successiva. Un modo poco elegante per aggirare questo problema è:

```
115 MOVE 180,22
125 MOVE 178,180
135 MOVE 20,178
run
```

... che produce l'effetto desiderato. Vi è tuttavia un modo più semplice che consiste nell'aggiungere un secondo parametro (,0) alla fine del comando MASK il quale chiede di non disegnare il primo punto di ogni linea. Si editi la linea 90 e si scriva tale parametro:

```
90 MASK &X00111100,0
```

... e si cancellino le linee aggiunte inserendo:

```
115
125
135
```

Ora si riesegua il programma e questa volta il riquadro tratteggiato sarà simmetrico. Si noti che se il secondo parametro di MASK è 1 il comando torna a disegnare tutte le linee a partire dal primo pixel.

Si guardi ora tra i tratteggi delle linee. Vi è qualcosa nel triangolo in basso a destra che non si trova nel triangolo in alto a sinistra. Questa è la carta della grafica che è stata posta uguale al colore 2 con il comando CLG 2 della linea 50 e che è invisibile nel triangolo in alto a sinistra perchè è dello stesso colore dello sfondo. Si modifichi ora la linea 50:

```
50 CLG 2:GRAPHICS PAPER 0
```

... e si riesegua il programma. La carta viene ora mostrata in modo chiaro in tutto il riquadro.

E' possibile rendere invisibile (o trasparente) la carta della grafica. Ciò significa che una linea tratteggiata disegnata su un'immagine preesistente manterrà ciò che si trova tra i punti. Il cursore grafico diventa cioè trasparente aggiungendo un ",1" al comando GRAPHICS PEN. Ritornerà non trasparente utilizzando il parametro ",0". Si modifichi la linea 70:

```
70 GRAPHICS PEN 1,1
run
```

... e si osservi il risultato.

Oltre a disegnare linee è possibile scrivere testi alla posizione del cursore grafico. In questo modo è possibile posizionare i caratteri con una precisione molto superiore (a passi di 1 pixel invece di 8) e che possiamo disegnare caratteri utilizzando i colori grafici (si veda più avanti).

Per scrivere caratteri alla posizione del cursore grafico, si porti il cursore grafico nell'angolo in alto a sinistra del primo carattere che deve essere stampato e poi si esegua il comando TAG (o TAG #1 ecc. per altri canali di testo) seguito da un normale comando PRINT. Il cursore grafico verrà spostato di 8 caratteri a destra per ogni carattere che viene stampato. Si aggiunga:

```
160 MOVE 64,108
170 TAG
180 PRINT "MARCO"
190 TAGOFF
run
```

Ogni messaggio di output di BASIC verrà inviato allo schermo di testo qualunque sia la modalità utilizzata (TAG/TAGOFF) ma è conveniente uscire dalla modalità TAG quando non se ne ha più bisogno.

Ma cosa sono quelle frecce dopo il nome? Bene, questi sono i caratteri di ritorno carrello CHR$(13) e di a capo CHR$(10). Il software di gestione della grafica trasforma tutti i primi 32 caratteri ASCII nelle corrispondenti versioni stampabili (come se fossero preceduti da un CHR$(1)). Ciò avviene perchè la maggior parte dei primi 32 caratteri ha un particolare significato solo per lo schermo di testo. Per lo stesso motivo, se si supera la larghezza dello schermo non viene effettuato alcun A capo automatico.

I caratteri di ritorno carrello e di a capo possono essere eliminati con la solita tecnica che consiste nel terminare la linea con un punto e virgola:

```
180 PRINT "MARCO";
run
```

Il testo inviato allo schermo grafico usando TAG subisce l'influenza dei comandi GRAPHICS PEN utilizzati per il tracciamento di linee. Perciò il nome viene scritto con il colore 1 (a causa di GRAPHICS PEN 1) ed è trasparente. Il comando:

```
150 GRAPHICS PEN 1,0
run
```

... tornerà ad usare la carta opaca, mentre:

```
150 GRAPHICS PEN 0,1
```

... scriverà con il colore 0, in modo trasparente.

Si cancelli ora la linea 150 e si esegua nuovamente il programma. La penna grafica ritornerà del colore 1 in modo trasparente (fissati alla linea 70).

Caratteri trasparenti

E' anche possibile aggiungere caratteri trasparenti allo schermo di testo usando un determinato codice di controllo. Si aggiunga:

```
200 PRINT #2,CHR$(22);CHR$(1)
210 LOCATE #2,32,14:PRINT #2,"******"
220 LOCATE #2,32,14:PRINT #2,"------"
230 PRINT #2,CHR$(22);CHR$(0)
run
```

La linea 200 pone il canale #2 in modo trasparente. Si noti che i caratteri di sottolineatura appaiono insieme agli asterischi. In questo modo è possibile costruire caratteri composti, anche in più colori. La linea 230 esce dal modo trasparente e riporta il canale #2 al modo opaco.

Colori e modalità

E' possibile disegnare usando una modalità che permette di mescolare il colore del disegno a quello che è già presente sullo schermo. Il colore finale viene calcolato effettuando una combinazione logica tra il colore che è presente sullo schermo e il colore (della penna o della carta) con cui si disegna. Le combinazioni logiche disponibili sono XOR, AND e OR. Questa modalità può essere specificata o come quarto parametro di DRAW/DRAWR, PLOT/PLOTR, MOVE/MOVER o "stampando" con PRINT la sequenza CHR$(23);CHR$(<modalità>). In entrambi i casi il valore 1 sta per XOR, 2 per AND e 3 per OR. Il valore 0 ripristina la normale modalità in cui il colore del disegno sostituisce quello già presente sullo schermo.

Il prossimo esempio mostra l'uso della combinazione XOR. Tale combinazione è molto usata nella grafica della tartaruga in quanto ripassando due volte la stessa immagine, essa viene riportata ai colori originali. Perciò si esegue due volte la routine di disegno del riquadro (linee 110 e 130) e due volte si esegue anche la stampa in testo e grafica TAG alle linee 170 e 190. I comandi FRAME permettono di aggiungere un ritardo che rende visibile l'effetto. Si noti l'uso, nella linea 90, dei comandi senza parametro iniziale. Ciò è perfettamente normale e lascia ai primi parametri i valori precedenti.

Il terzo parametro (,1) del comando MOVE alla linea 220 chiede di usare il colore 1 per la penna grafica, modificando il valore 3 chiesto alla linea 60. La modalità XOR viene scelta utilizzando il quarto parametro del comando DRAWR alla linea 230. Si noti che anche qui si è tralasciato un parametro.

Un particolare effetto può essere visualizzato eliminando il comando MASK della linea 90. Gli angoli del quadrato spariscono in quanto vengono disegnati due volte (alla fine di una linea ed all'inizio della successiva) e quindi vengono cancellati per effetto della modalità XOR.

```
10 REM modalità XOR
20 MODE 1:INK 2,10:INK 3,4
30 ORIGIN 440,100,440,640,100,300
40 WINDOW 1,26,1,25
50 CLG 2:GRAPHICS PAPER 0
60 DRAW 200,200,3
70 MOVE 2,0:FILL 3
80 ORIGIN 440,0,440,640,0,400
90 GRAPHICS PEN ,1:MASK ,0
100 FOR y=60 to 318 STEP 2
110 GOSUB 220
120 FRAME:FRAME
130 GOSUB 220
```

continua nella prossima pagina

```
140 NEXT
150TAG
160 FOR y=60 to 318 STEP 2
170 MOVE 96,y:PRINT CHR$(224);
180 FRAME:FRAME
190 MOVE 96,y:PRINT CHR$(224);
200 NEXT
210 END
220 MOVE 90,y,1
230 DRAWR 20,0,,1
240 DRAWR 0,20
250 DRAWR -20,0
260 DRAWR 0,-20
270 RETURN
run
```

Animazione

E' possibile produrre un effetto di animazione cambiando i colori associati agli inchiostri. Anche se il contenuto della memoria rimarrà uguale, sullo schermo apparirà un movimento. Un esempio è contenuto nel programma "Welcome" nel lato 4 dei dischi di sistema (si immetta RUN "disc" per vedere il programma dimostrativo). Lo scambio dei colori non è tuttavia sufficiente se le figure si sovrappongono. Il prossimo esempio usa la modalità OR per stampare sullo schermo i numeri da 1 a 4. La forma viene determinata esaminando il carattere stampato nell'angolo in fondo a sinistra e riproducendo ogni punto con grossi blocchi grafici. I numeri vengono scritti uno ad uno usando i colori 1,2,4 e 8 in modalità OR fissata in questo caso da una sequenza di caratteri di controllo alla linea 50.

Le linee da 160 in poi scambiano i colori secondo una formula matematica. I colori vengono determinati controllando un colore per volta e cercando la componente binaria che stiamo cercando. Ad esempio il numero 3 verrà disegnato con il colore 4 e quindi per visualizzare il numero 3 dobbiamo allocare un colore visibile a tutti gli inchiostri in cui il numero contiene il numero binario 4. Questi inchiostri sono:

4(0100), 5(0101), 6(0110), 7(0111), 12(1100), 13(1101), 14(1110), 15(1111)

In un'applicazione pratica, gli inchiostri che dovranno essere cambiati dovrebbero essere calcolati, e le linee da 180 a 200 dovrebbero essere sostituite da una routine più veloce.

```
10 REM animazioni
20 ON BREAK GOSUB 220
30 FOR i=1 TO 15:INK i,26:NEXT
40 m(1)=1:m(2)=2:m(3)=4:m(4)=8
50 MODE 0:PRINT CHR$(23);CHR$(3);:TAG
60 FOR p=1 TO 4
70 GRAPHICS PEN m(p),1
80 LOCATE #1,1,25:PRINT #1,CHR$(48+p);
90 FOR x=0 TO 7
100 FOR y=0 TO 14 STEP 2
110 IF TEST(x*4,y)=0 THEN 140
120 MOVE (x+6)*32,(y+6)*16:PRINT CHR$(143);
130 MOVE (x+6)*32,(y+7)*16:PRINT CHR$(143);
140 NEXT y,x,p
150 LOCATE #1,1,25:PRINT#1," ";
160 FOR p=1 TO 4
170 FOR i=1 TO 25:FRAME:NEXT
180 FOR i=0 TO 15
190 IF (i AND m(p))=0 THEN INK i,0 ELSE INK i,26
200 NEXT i,p
210 GOTO 160
220 INK 1,26
run
```

Sprite a piani di colore

Nell'esempio precedente abbiamo visto come, avendo disegnato grafici con i colori 1, 2, 4 e 8 può essere generato un effetto di animazione modificando i colori. Se vengono usati gli stessi inchiostri ma i colori vengono fissati su valori diversi, si può produrre un effetto completamente diverso conosciuto come "piani di colore":

```
10 montagne
20 DEFINT a-z
30 INK 0,1:INK 1,26
40 INK 2,6:INK 3,6
50 FOR i=4 TO 7:INK i,9:NEXT
60 FOR i=8 to 15:INKi,20:NEXT
70 MODE 0:DEG:ORIGIN 0,150:CLG:MOVE 0,150
80 FOR x=16 TO 640 STEP 16
```

continua nella prossima pagina

```
90 DRAW x,COS(x)*150+RND*100,4
100 NEXT
110 MOVE 0,0:FILL 4
120 cx=175:GOSUB 320
130 cx=525:GOSUB 320
140 SYMBOL 252,0,0,&C,&1F,&30,&7F,&FF
150 SYMBOL 253,0,6,&E,&F2,2,&F2,&FE
160 SYMBOL 254,0,&60,&70,&7F,&7F,&7F,&7F
170 SYMBOL 255,0,0,0,&F8,&EC,&FE,&FF
180 pr$=CHR$(254)+CHR$(255)
190 pl$=CHR$(252)+CHR$(253)
200 TAG:t!=TIME
210 FOR x=-32 TO 640 STEP 4
220 x2=((608-x)*2)MOD 640:hl=RND*10:hr=50*SIN(x)
230 GRAPHICS PEN 8,1:MOVE x,100+hr,,3:PRINT pr$;
240 GRAPHICS PEN 2,1:MOVE x2,115+hl,,3:PRINT pl$;
250 IF (TEST(x2-2,115+hl-12) AND 8)=8 THEN 380
260 IF TIME-t!<30 THEN 260
270 FRAME:t!=TIME
280 GRAPHICS PEN 7,1:MOVE x,100+hr,,2:PRINT pr$;
290 GRAPHICS PEN 13,1:MOVE x2,115+hl,,2:PRINT pl$;
300 NEXT
310 GOTO 210
320 MOVE cx,100
330 FOR x=0 TO 360 STEP 10
340 DRAW cx+SIN(x)*50+10*RND,100+COS(x)*25+10*RND,1
350 NEXT
360 DRAW cx,100:MOVE cx,90:FILL 1
370 RETURN
380 ENT -1,1,1,1
390 SOUND 1,25,400,15,,1,15
400 FOR y=100+hr TO -132 STEP -2
410 GRAPHICS PEN 7,1:MOVE x,y,,2:PRINT pr$;
420 GRAPHICS PEN 8,1:MOVE x,y-2,,3:PRINT pl$;
430 NEXT
440 GOTO 70
run
```

Per spiegare come funziona questo programma, dobbiamo visualizzare il colore INK in binario. Partendo dal valore più alto (15), tutti i colori INK che possiedono l'ottavo bit (da 15 a 8) sono posti nel colore azzurro. Tutti i colori INK che possiedono il quarto bit (da 7 a 4) sono posti a verde. I colori INK 2 e 3 che possiedono il bit 2 sono posti a rosso. Infine il colore 1 è posto a bianco. INK 0 rimane blu.

Sullo schermo i grafici subiscono un OR sul posto - linee 230 e 240. Il colore di ogni pixel viene determinato dal bit più significativo del risultato in quel punto. Per questo motivo, un'immagine sul piano più significativo oscurerà sempre un'immagine su un piano meno significativo ma lo sfondo rimane e può essere visualizzato nuovamente se l'immagine "più significativa" viene spostata. Il modo in cui è possibile spostare un'immagine consiste nel disegnarla usando la modalità AND con i colori numero 7,11,13 o 14 eliminando i precedenti colori 8,4,2,1 rispettivamente - linee 280 e 290.

Grafica ed uso della memoria aggiuntiva (solo 6128)

Per concludere questo capitolo, presentiamo un piccolo programma di disegno grafico che utilizza i 64K aggiuntivi di RAM del 6128.

```
10 'SCREEN DESIGNER by DAVID RADISIC
20 '  copyright (c) AMSOFT 1985
30 '
40 'Remember to RUN "BANKMAN" before running program!
50 '*****************************************************
60 '
70 ON ERROR GOTO 2740
80 DEFINT a-x
90 MODE 1:ch=127:cmnd=1:pn(0)=0:pn(1)=26:pn(2)=15:pn(3)
   =6:pn(4)=0:pn=1:norx=1:menu=1:zzz=HIMEM
100 DIM command$(22)
110 norx$(0)="Normal":norx$(1)="XOR   ":norx$(2)="Trans
    p":norx$(3)="XOR   "
120 RESTORE:READ cmnds$(1),cmnds$(2):cmnd$=CHR$(16)+CHR
    $(&7F)+cmnds$(1)+cmnds$(2)
130 READ cmno:FOR i=1 TO cmno:READ command$(i):NEXT
140 READ st$:IF st$<>"**" THEN cmnd$(cmnd)=st$:cmnd=cmn
    d+1:GOTO 140
150 WINDOW #0,1,40,1,3:PAPER #0,0:PEN #0,1:CLS #0
160 WINDOW #1,1,40,4,4:PAPER #1,3:PEN #1,1:CLS #1
170 ORIGIN 0,0,0,640,0,334
180 x=320:y=200:MOVE x,y
190 BORDER pn(4):FOR i=0 TO 3:INK i,pn(i):NEXT
200 MASK 255,0:PAPER 0:PEN 1:PAPER #1,3:PEN #1,1:GRAPHI
    CS PEN pn,norx
210 IF flag<>5 THEN 280
```

continua sulla prossima pagina

```
220 IF pn<2 THEN pnt$=CHR$(240):px=(pn+1)*13 ELSE IF pn
    <4 THEN pnt$=CHR$(241):px=(pn-1)*13 ELSE pnt$=CHR$(
    243):px=37
230 LOCATE px,2:PRINT pnt$;
240 LOCATE 1,1:PRINT USING"   PEN 0 : ##   PEN 1 : ##";
    pn(0);pn(1);
250 LOCATE 29,2:PRINT USING"Border : ##";pn(4)
260 LOCATE 1,3:PRINT USING"   PEN 2 : ##   PEN 3 : ##";
    pn(2);pn(3);
270 LOCATE px,2:PRINT " ";
280 LOCATE #1,1,1:PRINT#1,USING"X :####  Y :####     ";
    x;y;:PRINT #1,"Plot mode : ";norx$(norx+(undraw*2))
    ;" ";
290 IF flag=0 THEN GOSUB 2260
300 '
310 GOSUB 970
320 '
330 IF flag>0 THEN 390
340 IF i$="" THEN 390
350 cmnd=INSTR(cmnd$,i$):IF cmnd=0 THEN 390
360 IF cmnd=1 THEN CLG:x=320:y=200:GOTO 390
370 IF cmnd=2 THEN RUN 70
380 ON cmnd-2 GOSUB 1240,1410,1520,1640,1840,1860,1950,
    2020,2090,2120,2170,2200,2660,2660,2660,2660,2390,2
    330,2200
390 IF tx=0 AND ty=0 THEN 200
400 IF flag>0 THEN 440
410 GOSUB 630
420 GOSUB 680:FRAME:GOSUB 680
430 GOTO 200
440 MOVE tempx,tempy,pn,1
450 ON flag GOSUB 470,490,550,640
460 GOTO 200
470 PLOT x,y:GOSUB 630:PLOT x,y
480 RETURN
490 DRAW tempx+x,tempy:DRAW tempx+x,tempy+y
500 DRAW tempx,tempy+y:DRAW tempx,tempy
510 GOSUB 630
520 DRAW tempx+x,tempy:DRAW tempx+x,tempy+y
530 DRAW tempx,tempy+y:DRAW tempx,tempy
540 RETURN
550 MOVE tempx,tempy:DRAWR x,y
560 IF triside=0 THEN 580
570 DRAW tempxx,tempyy:DRAW tempx,tempy
```

continua sulla prossima pagina

```
580 GOSUB 630
590 MOVE tempx,tempy:DRAW tempx+x,tempy+y
600 IF triside=0 THEN RETURN
610 DRAW tempxx,tempyy:DRAW tempx,tempy
620 RETURN
630 x=x+tx:y=y+ty:RETURN
640 MOVE tempx,tempy:DRAW x,y
650 GOSUB 630
660 MOVE tempx,tempy:DRAW x,y
670 RETURN
680 ' draw and undraw cursor
690 IF flag=5 THEN RETURN
700 MASK 255,1
710 IF flag>1 THEN xx=tempx+x:yy=tempy+y ELSE xx=x:yy=y
720 IF flag=4 THEN xx=x:yy=y
730 IF flag=1 THEN xx=x:yy=y
740 IF undraw=1 THEN 820
750 GOSUB 790
760 MASK 255,0
770 IF i$=" " THEN GOSUB 2150:i$=""
780 RETURN
790 MOVE xx-4,yy,pn,1:DRAW xx+4,yy
800 MOVE xx,yy-4:DRAW xx,yy+4
810 MOVE xx,yy,,xorn:RETURN
820 nx=1:GOSUB 1220
830 FRAME:GOSUB 1220
840 IF i$=" " THEN nx=norx:GRAPHICS PEN pn,1:GOSUB 1220
850 i$=""
860 IF flag<>6 THEN 760
870 IF moved=0 AND j$<>"" AND (j$<CHR$(240) OR j$>CHR$(
    247)) THEN ch=ASC(j$):moved=1
880 IF moved=0 THEN RETURN
890 LOCATE 5,2
900 FOR i=ch-5 TO ch+5
910 PEN ABS(i<>ch)+1
920 ch$=CHR$(1)+CHR$(ABS(i+256)MOD 256)
930 IF ch=i THEN PRINT" "ch$" "; ELSE PRINT ch$;
940 NEXT
950 PEN 1:PRINT"   = "ch"  ";
960 GOTO 760
970 ty=0:tx=0:GOSUB 680:FRAME:GOSUB 680
980 IF INKEY(0)<>-1 OR INKEY(72)<>-1 THEN ty=16
990 IF INKEY(2)<>-1 OR INKEY(73)<>-1 THEN ty=-16
1000 IF INKEY(8)<>-1 OR INKEY(74)<>-1 THEN tx=-16
```

continua sulla prossima pagina

```
1010 IF INKEY(1)<>-1 OR INKEY(75)<>-1 THEN tx=16
1020 IF INKEY(21)<>-1 OR INKEY(76)<>-1 THEN tx=tx/8:ty=
     ty/8
1030 IF tx=0 AND ty=0 THEN moved=0 ELSE moved=1
1040 j$=INKEY$:i$=UPPER$(j$)
1050 IF (i$=" " OR i$=CHR$(13)) AND flag>0 THEN 1090
1060 IF flag=5 THEN 1120
1070 IF flag=6 THEN 1170
1080 RETURN
1090 ON flag GOSUB 1240,1410,1640,1860,1950,2020
1100 i$=""
1110 RETURN
1120 IF moved=0 THEN RETURN
1130 IF tx>2 THEN pn=(pn+1) MOD 5 ELSE IF tx<-2 THEN pn
     =ABS((pn<1))*5-1+pn
1140 IF ty>2 THEN pn(pn)=(pn(pn)+1) MOD 27 ELSE IF ty<-
     2 THEN pn(pn)=ABS((pn(pn)<1))*27-1+pn(pn)
1150 GRAPHICS PEN pn:PEN #1,pn
1160 tx=0:ty=0:BORDER pn(pn):RETURN
1170 IF tx<0 THEN ch=ABS(ch+255) MOD 256
1180 IF ty<0 THEN ch=ABS(ch+246) MOD 256
1190 IF tx>0 THEN ch=(ch+1) MOD 256
1200 IF ty>0 THEN ch=(ch+10) MOD 256
1210 tx=0:ty=0:RETURN
1220 TAG:MOVE xx-8,yy+6,pn,nx:PRINT CHR$(ch);:TAGOFF
1230 RETURN
1240 ' C
1250 IF flag=1 THEN 1290
1260 ro=1:GOSUB 2240
1270 tempx=x:tempy=y:flag=1
1280 RETURN
1290 IF tempx=x AND tempy=y THEN 1390
1300 PLOT x,y,,1
1310 tix=MAX(x,tempx)-MIN(tempx,x):tiy=MAX(y,tempy)-MIN
     (tempy,y)
1320 ti=SQR((tix↑2)+(tiy↑2))
1330 ORIGIN tempx,tempy
1340 PLOT 0,0,pn,0:MOVE 0,-ti
1350 FOR z=0 TO PI*2+0.01 STEP PI/(ti/2)
1360 DRAW SIN(z+PI)*ti,COS(z+PI)*ti,pn,norx
1370 NEXT z
1380 ORIGIN 0,0
1390 x=tempx:y=tempy:tempx=0:tempy=0:flag=0
1400 RETURN
```

continua sulla prossima pagina

```
1410 ' B
1420 IF flag=2 THEN 1470
1430 ro=2:GOSUB 2240
1440 tempx=x:tempy=y:flag=2
1450 x=0:y=0
1460 RETURN
1470 IF norx=1 THEN 1500
1480 MOVE tempx,tempy:DRAW tempx+x,tempy,,norx
1490 DRAW tempx+x,tempy+y:DRAW tempx,tempy+y:DRAW tempx
     ,tempy
1500 x=tempx:y=tempy:flag=0
1510 RETURN
1520 ' F
1530 ro=3:GOSUB 2240
1540 GOSUB 1620:IF i$=" " THEN 1600
1550 edgecol=VAL(i$)
1560 ro=4:GOSUB 2240
1570 GOSUB 1620:IF i$=" " THEN 1600
1580 filler=VAL(i$)
1590 MOVE x,y,edgecol:FILL filler
1600 flag=0:i$=""
1610 RETURN
1620 i$=INKEY$:IF (i$<"0" OR i$>"3") AND i$<>" " THEN 1
     620
1630 RETURN
1640 ' T
1650 IF flag=3 THEN 1700
1660 flag=3:ro=5:GOSUB 2240
1670 tempx=x:tempy=y
1680 x=0:y=0
1690 RETURN
1700 IF triside<>0 THEN 1770
1710 ro=6:GOSUB 2240
1720 MOVE 0,0,pn,1:GOSUB 590
1730 tempxx=tempx+x:tempyy=tempy+y:x=x/2:y=20
1740 triside=1
1750 GOSUB 550:GOSUB 590
1760 RETURN
1770 IF norx=1 THEN 1800
1780 MOVE tempxx,tempyy,,norx:DRAW tempx,tempy
1790 DRAW tempx+x,tempy+y:DRAW tempxx,tempyy
1800 tempxx=0:tempyy=0
1810 x=tempx:y=tempy:triside=0
1820 tempx=0:tempy=0:flag=0
```

continua sulla prossima pagina

```
1830 RETURN
1840 ' @
1850 norx=1:undraw=undraw XOR 1:RETURN
1860 ' L
1870 IF flag=4 THEN 1910
1880 ro=7:GOSUB 2240
1890 tempx=x:tempy=y:flag=4
1900 RETURN
1910 IF norx=1 THEN 1930
1920 MOVE tempx,tempy,,norx:DRAW x,y
1930 x=tempx:y=tempy:flag=0
1940 RETURN
1950 ' I
1960 IF flag=5 THEN flag=0:CLS:INK 3,tmpcol:INK pn,col:
     GOTO 1990
1970 CLS:flag=5:BORDER pn(pn)
1980 RETURN
1990 FOR i=0 TO 3:INK i,pn(i):NEXT:BORDER pn(4)
2000 IF pn=4 THEN pn=1
2010 CLS:RETURN
2020 ' A
2030 IF flag=6 THEN 2070
2040 tempx=0:tempy=0:CLS
2050 undraw=1:flag=6:norx=1:moved=1
2060 RETURN
2070 flag=0
2080 RETURN
2090 ' N
2100 norx=0
2110 RETURN
2120 ' E
2130 GRAPHICS PEN pn,0:TAG:MOVE xx-8,yy+6,,0:PRINT " ";
     :TAGOFF
2140 RETURN
2150 '<SPACE>
2160 PLOT x,y,pn,norx:RETURN
2170 ' X
2180 norx=1
2190 RETURN
2200 ' M
2210 menu=menu MOD 2+1
2220 GOSUB 2260:RETURN
2230 i$=UPPER$(INKEY$):IF i$="" OR INSTR(ser$,i$)=0 THE
     N 2230 ELSE RETURN
```

continua sulla prossima pagina

```
2240 CLS:undraw=0:PRINT cmnd$(ro);:LOCATE 1,3:PRINT"<SP
     ACE> ";:IF ro=3 OR ro=4 THEN PRINT"To exit"
2250 RETURN
2260 CLS:flag=-1
2270 FOR i=1 TO LEN(cmnds$(menu))
2280 ps=i+ABS(menu=2)*LEN(cmnds$(1))
2290 PEN 1:PRINT"<"MID$(cmnds$(menu),i,1)">"MID$(comman
     d$(ps),2,4)" ";
2300 NEXT
2310 PRINT"<CLR>   <DEL>   <SPACE>";
2320 RETURN
2330 ' S
2340 GOSUB 2460:IF filename$="" THEN 2370
2350 GOSUB 2550
2360 SAVE filename$,b,&C000,&4000
2370 GOSUB 2260
2380 RETURN
2390 ' R
2400 GOSUB 2460:IF filename$="" THEN 2440
2410 GOSUB 2730
2420 LOAD filename$,&C000
2430 GOSUB 2570
2440 GOSUB 2260
2450 RETURN
2460 CLS:LOCATE 10,3:PRINT"<RETURN> to Abort!";
2470 LOCATE 1,1:PRINT"Enter Filename :";
2480 INPUT "",filename$:IF filename$="" THEN RETURN
2490 n=INSTR(filename$,"."):IF n=0 THEN 2520
2500 IF n=1 THEN 2460
2510 filename$=LEFT$(filename$,n-1)
2520 filename$=LEFT$(filename$,8)+".scn"
2530 CLS
2540 RETURN
2550 FOR i=0 TO 4:POKE &C000+i,pn(i):NEXT
2560 RETURN
2570 FOR i=0 TO 4:pn(i)=PEEK(&C000+i) MOD 27:NEXT
2580 cn=0:FOR i=0 TO 2:IF pn(i)=pn(i+1) THEN cn=cn+1
2590 NEXT:IF cn=3 THEN 2630
2600 FOR i=0 TO 3:INK i,pn(i):NEXT
2610 BORDER pn(4):pn=1:GRAPHICS PEN pn
2620 RETURN
2630 pn(0)=0:pn(1)=26:pn(2)=15:pn(3)=6:pn(4)=0
2640 GOTO 2600
2650 ' 1, 2, 3, & 4
```

continua sulla prossima pagina

```
2660 CLS:PRINT"Do you wish to <S>tore":PRINT TAB(16)"<R
     >etrieve":PRINT TAB(13)"or <E>xchange  the screen
     ?"
2670 ser$="SRE"+CHR$(13):GOSUB 2230:IF i$=CHR$(13) THEN
      2260
2680 bnk2=(cmnd-13):bnk1=1
2690 IF i$="S" THEN CLS:GOSUB 2550:|SCREENCOPY,bnk2,bnk
     1
2700 IF i$="R" THEN GOSUB 2730:|SCREENCOPY,bnk1,bnk2:GO
     SUB 2570
2710 IF i$="E" THEN CLS:GOSUB 2730:GOSUB 2550:|SCREENSW
     AP,bnk2,bnk1:GOSUB 2570
2720 GOSUB 2260:RETURN
2730 FOR i=0 TO 3:INK i,0:NEXT:BORDER 0:RETURN
2740 CLS:GOSUB 2600:RESUME 2260
2750 DATA "CBFT@LIANEXM","1234RSM"
2760 DATA 19,Circle,"Box  ","Fill ",Triangle,Alternate,
     "Line ","Inks ",ASCII,Normal,Erase,"Xor  ","Menu "
     ,"1st  ","2nd  ","3rd  ","4th  ",Restore,"Save ","
     Menu "
2770 DATA Circle,Box,Edge colour,Filler colour,Triangle
      1,Triangle 2,Line,**
```

I due comandi RSX, |SCREENCOPY e |SCREENSWAP usati in questo programma vengono forniti dal programma di utilità "Bank Manager" sul lato 1 dei dischi di sistema. I comandi facilitano la copia e lo scambio dei vari schermi tra i vari blocchi di memoria e tra i due banchi da 64K.

E' necessario perciò eseguire il programma "Bank Manager" PRIMA di eseguire questo programma e per fare ciò occorre inserire il lato 1 dei dischi forniti e scrivere:

run "bankman"

Poi si potrà eseguire il programma presentato.

Ed ora cosa accade?

Avendo eseguito il programma Bank Manager seguito dal programma Screen Designer presentato, apparirà un menù di opzioni ed un cursore grafico lampeggiante nel centro dello schermo.

Da ora in poi, si prema la lettera appropriata (ad esempio "C" per Circle) per eseguire la funzione desiderata.

Come disostrazione, si prema:

C

... poi si prema il tasto cursore con la freccia verso l'alto per portare il cursore grafico a 2 o 3 centimetri dal centro dello schermo.

Infine si prema la barra spaziatrice per eseguire la funzione Circle che come si sarà immaginato disegna un cerchio. Si tornerà così al menù iniziale.

Premendo M (per Menù) si passerà ad un altro menù da dove si può salvare (Save) o ripristinare (Restore) un determinato schermo e dove è possibile manipolare il contenuto degli schermi da 1 a 4 (contenuti nel secondo banco di 64K di memoria).

Per usare queste funzioni, si deve immettere il "numero della memoria" (1, 2, 3 o 4) ed in tal modo verrà visualizzato un altro menù.

Da qui si può selezionare:

S Per memorizzare uno schermo

R Per ripristinare uno schermo

E Per cambiare schermo

[RETURN] Per uscire dal menù

Se ad esempio si vuole memorizzare lo schermo attuale nella memoria numero 2, si prema 2 seguito da S.

Ritornando all'altro menù (dopo un ripristino, un salvataggio o una manipolazione dello schermo), premendo M si ritornerà al menù iniziale.

Il programma Screen Designer permette di effettuare molte manipolazioni grafiche tra le quali vi sono rettangoli, cerchi, triangoli, linee, punti, riempimenti e disegno di caratteri.

Quando uno schermo è stato completato, può essere salvato su disco usando l'opzione Save (Salva) per poi essere ricaricato con Restore (Ripristina).

Ora si conclude l'ultimo capitolo del manuale. Usando, facendo esperimenti e analizzando i programmi presentati si dovrebbe aver acquisito una discreta padronanza del BASIC AMSTRAD e del CPC

Ulteriori dettagli ...

Per concludere presentiamo 4 appendici che comprendono: la licenza d'uso di CP/M, un glossario dei termini, alcuni listati di giochi ed un indice del manuale.

Speriamo che abbiate gradito questo manuale e ringraziamo per la fiducia accordataci acquistando il CPC.

Appendice 1
AMSTRAD

Accordo di licenza d'uso del software

Licenza relativa al software descritto in questo manuale.

MOLTO IMPORTANTE

Amstrad fornisce il Software ed il Programma descritto in questo manuale incorporato o contenuto su dischi o nastri magnetici, su disco rigido o su altro supporto e il materiale relativo al Programma in questo manuale o in altri documenti soggetti alle condizioni di questa Licenza d'uso.

Rompendo il sigillo sulla confezione che contiene il Software e/o accedendo, utilizzando, eseguendo, listando, copiando o manipolando in altro modo il disco, il nastro, il disco rigido o altro tipo di supporto, il Programma, il manuale o altra documentazione per motivi diversi dalla cancellazione o distruzione come indicato di seguito, si accettano tutte le clausole e condizioni di questa Licenza d'uso.

Se non si accettano una o più termini o condizioni di questa Licenza d'Uso, si dovrà distruggere la confezione contenente il software, i manuali e i documenti e cancellare il Programma ed il Software dal disco, nastro, disco fisso o altro supporto su cui viene fornito.

Definizioni

1. Con "Macchina" si intende un singolo microcomputer o PC sul quale si usa il Software. I sistemi con più di una unità centrale di elaborazione (CPU) richiedono Licenze aggiuntive.

2. Con "Software" si intende l'insieme dei programmi, dischi, nastri, documentazione ed altro materiale connesso, contenuti nella Confezione o indicati dal Manuale.

3. Con "Programma" si intende una parte del software e cioè istruzioni, codice, messaggi ed altre informazioni contenute nei dischi o nei nastri magnetici, su un disco fisso o su un altro supporto o su parte di esso.

4. Con "Amstrad" si intende Amstrad SpA, via Riccione 14, Milano.

5. Con "Licenziatario" si intende l'acquirente che ha acquistato per proprio uso oppure l'azienda e i suoi dipendenti quando l'acquisto è stato effettuato per uso aziendale.

Licenza

Amstrad concede al Licenziatario una Licenza non esclusiva per il Software per:

1. Usare con una sola Macchina il Programma fornito.

2. In mancanza di protezione contro le copie, copiare il Software in qualsiasi forma leggibile dalla macchina per la copia di riserva o per le modifiche del Licenziatario richieste dall'uso del Software su una sola Macchina. Ogni porzione del Software copiata in questo modo continua ad essere soggetta alle clausole di questa Licenza. Il Licenziatario può effettuare solo una (1) copia del Software in forma leggibile dalla Macchina per questi scopi. E' vietato copiare la documentazione e altro materiale stampato. E' vietato disassemblare, decompilare o comunque manipolare il Programma o il Software.

Clausole

1. Il Software fornito da Amstrad è soggetto a diritti d'autore. Il Licenziatario riconosce ed accetta questi diritti d'autore.

2. Questo Accordo di Licenza entra in vigore all'apertura o alla rottura dei sigilli o quando si accede, usa, esegue, lista, copia o si manipola in altro modo il Programma e ha effetto fino alla sua recisione.

3. Il Licenziatario può rescindere questo Accordo di Licenza in ogni momento distruggendo il Software con tutte le sue copie. I benefici derivanti al Licenziatario dalla Licenza hanno termine automaticamente se vengono violate le clausole di questo Accordo e Amstrad può richiedere la restituzione immediata del Software. In nessun caso la rescissione pregiudica alcun diritto acquisito dalle parti.

4. La Licenza è personale del Licenziatario e non può essere ceduta ad altre persone od aziende.

5. Al Licenziatario non è concesso il diritto di "prestare", "affittare", "vendere" o trasferire altrimenti il Software in parte o nella sua totalità.

6. Il Licenziatario non può usare, copiare, modificare o trasferire in parte o nella sua totalità il Software, oppure una sua copia parziale o totale, integra o modificata, neppure inclusa in altro materiale, a meno che non sia espressamente previsto in questo Accordo di Licenza.

7. Il Software non può essere trasferito mediante supporti diversi da quelli su cui è stato fornito. Non può pertanto essere trasferito mediante supporti quali le linee di telecomunicazione.

8. Il Licenziatario si impegna a prendere ogni possibile provvedimento per proteggere il Software da uso, riproduzione o distribuzione non autorizzati.

Garanzia limitata

IL SOFTWARE VIENE FORNITO "TAL QUALE". IL LICENZIATARIO SI ASSUME LA RESPONSABILITA' DELLA SCELTA DEL PROGRAMMA PER RAGGIUNGERE DETERMINATI RISULTATI ED IL LICENZIATARIO E' RESPONSABILE DELL'INSTALLAZIONE, DELL'USO E DEI RISULTATI DERIVANTI DALL'USO. AD ECCEZIONE DI QUANTO ESPRESSAMENTE STABILITO IN QUESTO ACCORDO, AMSTRAD NON INTENDE NE' OFFRE ALCUNA GARANZIA DI ALCUN GENERE, IMPLICITA OD ESPLICITA, INCLUSE LE IMPLICITE GARANZIE, SENZA LIMITAZIONE ALCUNA, DI COMMERCIABILITA' O DI ADEGUATEZZA A SCOPI SPECIFICI O GENERICI.

AMSTRAD NON GARANTISCE CHE LE FUNZIONI CONTENUTE NEL SOFTWARE SODDISFINO LE ESIGENZE DEL LICENZIATARIO, NE' CHE IL FUNZIONAMENTO DEI PROGRAMMI SIA INTERROTTO O ESENTE DA ERRORI.

AMSTRAD NON RACCOMANDA L'USO DEL SOFTWARE PER FUNZIONI CRITICHE O PER APPLICAZIONI DOVE UN ERRORE DEL SOFTWARE POSSA CAUSARE PERDITE FINANZIARIE, DANNI O SPESE.

Tuttavia AMSTRAD garantisce che i dischi, i nastri od ogni altro supporto nel quale è accluso il programma sono esenti da difetti materiali e di fabbricazione e che rimarranno tali in condizioni di uso normale per un periodo di novanta (90) giorni dalla data di ricevimento del licenziatario, certificata dalla copia del documento di acquisto del licenziatario. nell'eventualità che dischi, nastri o supporti presentino tali difetti nel suddetto periodo, AMSTRAD li sostituirà gratuitamente, se restituiti in porto franco.

Amstrad inoltre A SUA DISCREZIONE, produrrà ogni sforzo per porre rimedio, con modifiche o alterazioni, ad ogni errore di programma, attribuibile ad Amstrad, purchè sia dimostrabile, provato e notificato in forma scritta ad Amstrad entro il suddetto periodo.

Limitazioni

IN NESSUN CASO IL FABBRICANTE, LO SVILUPPATORE O IL FORNITORE DEL SOFTWARE SOTTO LICENZA E' IMPUTABILE PER DANNI O ALTRE PERDITE ECONOMICHE O FISICHE, CONSEGUENTI, INCIDENTALI, INDIRETTE, SPECIALI O PARTICOLARI, ANCHE SE AMSTRAD FOSSE PREAVVERTITA DELLA POSSIBILITA' DI DANNI O PERDITE. IN NESSUN CASO L'INDENNIZZO PUO' ECCEDERE IL PREZZO D'ACQUISTO DEL SOFTWARE.

Indennizzo

Il Licenziatario ha l'obbligo di risarcire Amstrad per ogni obbligo assunto da Amstrad derivante da azioni legali o giudiziarie intraprese dal Licenziatario direttamente contro Amstrad o contro terzi, per negligenza o rottura di contratto.

Termini di legge

Il presente Accordo ha applicazione, interpretazione e vigore nel rispetto delle leggi vigenti.

Restrizione dei diritti per le leggi U.S.A. nei riguardi del software Microsoft

Il programma e la documentazione sono soggetti ad una restrizione dei diritti. L'uso e la duplicazione è soggetto a restrizioni come stabilito nelle suddivisioni b(3)(ii) del "The Rights in Technical Data and Computer Software clause" del 252.227-7013 del DOD-FAR.

Il contraente/produttore è Microsoft Corporation / 160 11 NE 36th Way / Box 9701 / Redmond, WA 98073-9717

Termini generali

Se una delle norme sopra riportate o parte di essa risultasse in contrasto con le leggi vigenti, quella norma o la sua parte soltanto dovrà considerarsi nulla.

ALCUNI STATI NON PERMETTONO L'ESCLUSIONE DELLE GARANZIE IMPLICITE O CONDIZIONI E LIMITAZIONI DI RESPONSABILITA'. IN TALI CIRCOSTANZE LE LIMITAZIONI ED ESCLUSIONI NON VERRANNO ATTRIBUITE AL LICENZIATARIO.

Esportazione di tecnologia U.S.A.

Non è possibile esportare, vendere o trasferire direttamente o indirettamente a persone che possano esportare Programma o Software senza applicare ed ottenere la licenza di esportazione o di ri-esportazione dal Governo U.S.A. emesso dal "US Department of Commerce", Washington DC 20230

AMSTRAD SpA,
via Riccione, 14
20141 - MILANO

Appendice 2
GLOSSARIO DEI TERMINI

Alcuni dei termini usati più comunemente in ambiente informatico spiegato agli utenti del CPC ....

A/D
Analogico
Uno stato in cui la modifica tra l'inizio e la fine di un punto accade gradualmente invece che con passi istantanei. I computer sono dispositivi digitali; il mondo naturale è basato su principi analogici quindi, il computer deve eseguire una conversione da analogico a digitale (A/D) prima di poter processare i dati di una sorgente analogica.

Accoppiatore acustico
Noto anche come Modem acustico. E' uno strumento elettronico che serve a connettere un telefono al computer ed abilita quest'ultimo a comunicare mediante la normale linea telefonica. In tal modo il computer può comunicare con un servizio di informazione pubblica come Videotel, o con altri utenti per scambiare software ed ottenere dati.

Accumulatore
Una locazione di memoria all'interno del microprocessore, che memorizza temporaneamente i dati mentre vengono elaborati. Sono molto usati nella programmazione in codice macchina; non è necessario per gli utenti di BASIC essere a conoscenza della loro esistenza.

Alfabetizzazione informatica
Un'altra espressione pomposa che significa conoscenza dei computer.

Alfanumerico
Gli attributi che descrivono la differenza tra lettere o numeri e caratteri grafici.

Algebra booleana
E' una istruzione di relazione logica che può assumere solo due forme: vera o falsa. Questi ultimi sono denotati di solito 0 o 1.

Algoritmo
Un nome pomposo per una formula complicata. E' una sequenza di passi aritmetici e logici al fine di eseguire un calcolo.

Alimentatore
Permette di utilizzare la corrente di una normale presa per alimentare il computer.

ALU
Arithmetic Logic Unit (Unità Logico Aritmetica) E' la parte di microprocessore che esegue le operazioni aritmetiche e logiche: non riguarda l'utente direttamente ad eccezione dei programmatori in linguaggio macchina.

AMSDOS
AMStrad Disc Operating System o sistema operativo a dischi AMSTRAD. E' il programma che permette al BASIC Locomotive di accedere ai file su disco.

Animazione
I cartoni animati sono la forma meglio conosciuta di animazione; l'animazione su un computer è basata sul concetto di movimento di elementi grafici per simulare l'idea del movimento 'reale'.

Architettura
Il modo particolare di un computer di collegare una periferica alla CPU.

Archivio
Un insieme di qualsiasi tipo di dati in un formato indirizzabile dal computer.

Argomento
Una variabile indipendente, ovvero nell'espressione x+y=z, i termini x,y e z sono argomenti.

ASCII
American Standard Code for Information Interchange. Il modo più comunemente usato per rappresentare numeri, lettere ed altri simboli che possono essere inseriti dalla tastiera del computer o richiamati usando altri comandi.

Assembler
Un metodo di programmazione in cui le istruzioni in linguaggio macchina vengono richiamate in modo mnemonico (lettere che suggeriscono la funzione che deve essere eseguita dalla corrispondente routine in linguaggio macchina).

Avviamento "a caldo"
Viene eseguito mediante [CTRL]C premuto durante l'esecuzione di CP/M. Tale avviamento reinizializza il sotto-sistema del disco e fornisce il controllo nuovamente al CP/M.

Avviamento "a freddo"
Il processo di avviamento ed inizializzazione di un sistema operativo. Viene operato mediante il comando |CPM.

Base
L'aspetto numerico più importante per qualunque matematico. La base di ogni rappresentazione numerica. Il sistema binario ha base 2; il sistema decimale ha base 10 e quello esadecimale ha base 16.

BASIC
Beginner's All-purpose Symbolic Instruction Code. Un linguaggio di programmazione interpretativo usato in quasi tutti i computer domestici. BASIC è stato progettato in modo specifico per essere semplice da apprendere e facile da usare poichè permette ai programmi di essere 'uniti' tra loro e testati (eseguiti) a qualsiasi stadio del loro sviluppo al contrario dei compilatori nei quali il programma, prima di essere eseguito, deve essere stato completato.

Baud
Un bit per secondo: è l'unità di misura della velocità alla quale un segnale digitale viene trasmesso in sistemi di comunicazione seriali.

BCD
Binary Coded Decimal. Un sistema di codifica per numeri decimali nel quale ogni cifra viene rappresentata come un gruppo di quattro cifre binarie.

BDOS
Basic Disc Operating System - E' la parte del CP/M che permette ad un programma utente di utilizzare le possibilità del CP/M.

Binario
Il sistema numerico di base 2 nel quale tutti i numeri sono composti dalle cifre 0 e 1 (vedere la Parte 1 del capitolo intitolato "A vostra disposizione ... ").

BIOS
Basic Input Output System. Questa è la parte del CP/M che dipende dalla macchina e che viene preparato in modo specifico per un dato tipo di computer. Tutto l'Input e l'Output riguardanti lo schermo, la tastiera, il disco ecc. viene gestito dal BIOS.

Bit
Abbreviazione di BInary digiT (ovvero cifra binaria). Una cifra binaria è composta da 0 e 1 e sono usati nel sistema numerico binario.

Bit meno significativo
In un numero binario il bit meno significativo (Least Significant Bit o LSB) è il bit all'estrema destra di una espressione.

Bit più significativo
In un numero binario il bit più significativo (Most Significant Bit o MSB) è il bit all'estrema sinistra di una espressione.

Boot
Il processo di caricamento del sistema operativo. Quando si avvia il CP/M da BASIC, dal disco viene caricato un breve programma di avvio che a sua volta carica il resto del sistema operativo.

Buffer
Un'area di memoria passeggera contenente informazioni nel corso di un trasferimento di queste da una parte all'altra del sistema, ad esempio da una cassetta all'unità centrale del computer (CPU) e nella RAM. Un buffer regola il modo con cui i dati vengono inviati al dispositivo operante ad una velocità diversa come, ad esempio, un modem o una stampante.

Bug
Un problema che varia da un 'comportamento inaspettato' basato su alcuni aspetti oscuri di un programma (come ad esempio se si premono quattro tasti contemporaneamente si modifica il colore dello schermo) ad una sequenza che rovina irrevocabilmente e completamente un programma o cancella la memoria di dati.

Bus

Un gruppo di connessioni contenute all'interno del computer o che lo collegano al mondo esterno che trasportano informazioni sullo stato della CPU, della memoria RAM ed altre informazioni o caratteristiche hardware. I segnali del bus del 464/6128 si trovano su un pettine situato sul retro della tastiera.

Byte
Un gruppo di otto bit, che formano la parte più piccola della memoria che una CPU ad otto bit possa richiamare e memorizzare.

CAD
Computer Aided Design (progettazione assistita dal computer). Una interazione di potenza di calcolo e grafica al fine di fornire una tavola di disegno elettronica, sebbene qualsiasi calcolo eseguito su un computer nel corso di un progetto prende il nome di CAD.

CAE
Computer Aided Education (educazione assistita dal computer). L'aiuto del computer per l'educazione. CAI (Computer Aided Instruction ovvero istruzione assistita dal computer) e CAI (Computer Aided Learning ovvero apprendimento assistito dal computer) sono due aspetti del CAE.

Canale
La via usata dal computer per effettuare un output: lo schermo, la stampante o una cassetta.

Carattere
Un qualsiasi simbolo che può essere rappresentato e visualizzato dal computer, incluso lettere, numeri e simboli grafici.

Carattere grafico
Una forma o un modello che può servire a creare immagini.

Cartuccia
Uno speciale circuito integrato contenente software che può essere inserito direttamente nel computer mediante una presa specifica. Il software fornito su cartuccia viene caricato ed eseguito più velocemente delle normali cassette, ma il suo costo è notevolmente superiore.

Cassette
Oltre alle ovvie cassette di musica è un termine generico che racchiude una gamma di 'pacchetti' incluso il software ROM, ecc.

Cattura di dati
Il termine che descrive la raccolta di dati da una sorgente esterna che è collegata in qualche modo ad un computer centrale.

CCP
Console Command Processor. E' una parte del CP/M che interpreta ed esegue l'input ricevuto dalla tastiera.

Chip
Un ingannevole ma popolare riferimento a qualsiasi forma di circuito integrato elettronico. Il 'chip' è in effetti un piccolo rettangolo di materiale silicio con il quale il circuito è costruito.

Ciclo
Un processo in un programma che viene eseguito ripetutamente da computer finchè una condizione viene soddisfatta.

Circuito integrato
Un insieme di circuiti elettronici miniaturizzati ed inseriti in un unico pezzo di silicio. Vedere anche chip.

Codice
A prescindere dalle implicazioni letterali, tale termine è usato frequentemente come abbreviazione di 'codice macchina'.

Codice a barre
Si guardi ad esempio sotto la confezione di una saponetta. Un computer che legge codici a barre può leggere mediante tecniche ottiche come la scansione mediante un laser a bassa energia.

Codice macchina
Il linguaggio di programmazione direttamente comprensibile da microprocessore, i cui comandi sono composti da cifre binarie.

Comandi interni
Comandi che appartengono al sistema operativo. Sono più veloci dei programmi transienti in quanto non vengono caricati da disco.

Comando
Una istruzione di programmazione.

Compilatore
Un programma che converte programmi scritti in un linguaggio interpretativo ad alto livello come il BASIC in istruzioni in codice del microprocessore, in tal modo le operazioni vengono eseguite ad una velocità molto maggiore.

Computer della quinta generazione
Sono soprattutto grossi computer che si pensa possano giungere ad autoprogrammarsi usando gli sviluppi dell'intelligenza artificiale.

CP/M
Sistema operativo standard della Digital Research che fornisce un'interfacia standard a programmi scritti per un'ampia gamma di sistemi a microprocessore.

CPU
Central Processing Unit (Unità centrale di elaborazione). Il componente nel cuore di qualsiasi computer che interpreta le istruzioni e fa sì che quest'ultimo le esegua; in un microcomputer, la CPU è il microprocessore stesso.

Cursore
Un indicatore spostabile che indica il luogo, sullo schemo, in cui apparirà il carattere successivo.

Cursore grafico

E' simile al cursore del testo, ma viene utilizzato nello schermo grafico. E' un elemento invisibile ma nonostante ciò è indispensabile per fare disegni. Non deve essere confuso con i caratteri grafici che appartengono al set di caratteri e vengono normalmente stampati alla posizione del cursore del testo.

Debugging
L'eliminazione degli errori in un programma mediante una combinazione di tentativi e metodi più scientifici.

Diagnostica
Un messaggio prodotto automaticamente dal computer per indicare e identificare un errore in un programma.

Diagramma di flusso
Una rappresentazione dei passi di un programma che indica la sequenza degli eventi che avvengono durante l'esecuzione di un programma.

Digitale
Descrive l'espressione di modifica di quantità in termini di passi discreti piuttosto che continui. E' l'opposto di analogico.

Digitalizzatore
Un dispositivo per inserire informazioni analogiche in un computer. Si fa riferimento di solito in unione con le tavolette grafiche.

Direttrice
Una parte di disco che contiene alcune informazioni relative ad ogni file del disco. E' un elenco di ciò che è contenuto nel disco.

Disco
Un disco di plastica piatto e circolare rivestito su uno o entrambi i lati con una superficie magnetica ed usato come mezzo di memorizzazione dei dati. Il disco è sempre situato in una busta di protezione quadrata di carta o di plastica; ad esso si accede mediante una finestra di lettura situata sulla busta di protezione.

Documentazione
I manuali forniti con il computer o il software che spiega il suo uso.

Doppia faccia
Un disco che può memorizzare informazioni su entrambi i lati. Un drive per dischi a doppia faccia permette di utilizzare entrambe le facce del disco senza girarlo.

DOS
Disk Operating System. Il software che controlla tutte le operazioni di un drive; si comporta come un sorvegliante nel 'parcheggio' rappresentato dalla disposizione logica del disco.

Drive
L'unità che registra le informazioni sulla superficie magnetica di un disco e 'legge' le informazioni registrate.

Dr. LOGO
La versione di LOGO (un linguaggio che utilizza la "tartaruga grafica") prodotto dalla Digital Research.

Editing
Correzione di dati, di un programma o di testi.

Editor
Un programma che si trova normalmente nella ROM del computer e che permette il processo di editing.

Editor di schermo
Un editor di testi o programmi in cui il cursore può essere spostato sullo schermo.

EPROM
Erasable Programmable Read Only Memory. Simile a una PROM ma i dati contenuti nel chip possono essere cancellati usando raggi ultravioletti e quindi riprogrammati. Una EEPROM è simile ma viene cancellata elettricamente.

Errore di sintassi
Si presenta se viene infranta una regola sull'uso delle parole chiave o delle variabili. Il BASIC presenterà il messaggio Syntax error.

Espressione
Una formula semplice o complessa usata in un programma per eseguire un calcolo. L'espressione definisce normalmente la natura dei dati che può utilizzare. In Dr. LOGO una espressione consiste di un nome di procedura seguito dagli input della procedura stessa.

File
Una raccolta di informazioni, normalmente memorizzata su cassetta o disco. Alcuni computer possono usare anche file su RAM.

File Esadecimale
Una rappresentazione in ASCII di un file di comandi o di un file in codice macchina.

Firmware
Software contenuto in ROM: un punto di incontro tra puro software e puro hardware.

Floppy
Un disco magnetico estraibile di 5.25' o di 8' di diametro utilizzato per memorizzare i dati. E' contenuto in una busta protettiva; ha una capacità molto superiore a quella di una cassetta. E' molto più veloce e più costoso.

Foglio elettronico
Un programma che permette di riempire e manipolare righe e colonne di numeri. Modificando un numero cambiano tutti quelli che da esso dipendono.

Forth
Un linguaggio di programmazione ad alta velocità con una velocità e una complessità che si trova tra un linguaggio ad alto livello e il linguaggio macchina: non è un linguaggio per principianti.

Generatore sonoro
La parte hardware e software di un computer, dedicata alla produzione di suoni.

Generazioni di computer
Le nuove scoperte tecnologiche hanno prodotto diverse fasi di sviluppo nel campo dei computer. In tal modo si sono prodotte diverse generazioni di computer a seconda delle varie tecnologie usate per costruirli.

Giochi di avventura
Un culto per alcuni e una noia per altri. Giochi basati su testi nei quali l'utente che gioca è invitato a partecipare in una serie di eventi pseudo-casuali basati sulla ricerca di un via di uscita in un labirinto.

Giochi d'azione
E' il tipo di gioco per computer dove: gli uomini spaziali invadono e vengono conquistati, mostri insaziabili inseguono in un labirinto e si impadroniscono di un incauto, personaggi sotto il controllo dell'utente cercano di evitare in tutti i modi una spiacevole 'morte'. Sono divertenti e permettono di sviluppare buoni riflessi ma sono di scarsa utilità per gli studenti informatici.

Grafica
La parte dello schemo del computer che non è in relazione alla visualizzazione dei caratteri ma al disegno di linee, di cerchi, di grafici ecc. Con una stampante appropriata è possibile anche avere copie dello schermo su carta.

Grafica della tartaruga
L'immagina grafica lasciata sullo schermo dal movimento della tartaruga. Man mano che la tartaruga si muove, lascia una scia sullo schermo.

Handshaking
Una sequenza di segnali elettronici che inizia, controlla e sincronizza lo scambio di dati tra un computer e una periferica o tra due computer.

Hardcopy
Stampa su carta di un programma, di altri testi o di uno schermo grafico. In questo caso lo schermo viene chiamato softcopy.

Hardware
La parte elettronica e meccanica di un computer; tutto ciò che non è software o firmware.

I/O
Input/Output.

IEEE-488
Una delle interfacce standard per collegare i dispositivi ad un microcomputer. E' simile ma non completamente compatibile con l'interfaccia parallela Centronics.

Indirizzo
E' il numero in una istruzione che identifica la locazione di una 'cella' nella memoria del computer. Mediante tale indirizzo è possibile selezionare una locazione di memoria particolare in modo sia possibile trovare e leggere il suo contenuto o, nel caso della RAM leggervi e scrivervi dopo eventuali modifiche.

Ingegneria del software
Indica la programmazione di un computer, utilizzando tecniche precise e non un approccio intuitivo.

Inizializzazione
Accensione di un sistema e dichiarazione di particolari valori delle variabili prima di iniziare l'esecuzione di un corpo di un programma; ad esempio la dichiarazione di una variabile intera, ecc.

Input
Tutto ciò che entra nella memoria del computer dalla sua tastiera, dalla sua unità a cassette, dal suo disco, dall'interfaccia seriale o da un altro dispositivo.

Insieme di istruzioni
Il processo logico e matematico più importante eseguito da un microprocessore. Ogni istruzione di alto livello (compresi i codici mnemonici dell'assembler) devono essere tramutati in istruzioni comprensibili dal computer. Un solo comando ad alto livello può provocare l'esecuzione di molte istruzioni dell'insieme compreso dalla CPU.

Intelligenza artificiale AI
Tecniche di programmazione che permettono al programma di apprendere dalle esperienze passate.

Interattivo
Si riferisce normalmente a programmi in cui il computer chiede all'utente di fornire vari tipi di input che possono servire a controllare una nave spaziale in un gioco di azione o a rispondere a domande in un programma educativo. L'azione dell'utente avrà effetto in tempo reale, cioè durante l'esecuzione del programma.

Interfaccia

E' una porta di accesso al computer sia in termini elettrici che umani. Le interfacce di cui è dotato il computer sono la tastiera (input), lo schermo (output) e tutte le prese che si trovano sul retro e che servono a collegare varie periferiche al computer.

Interfaccia parallela

Interfaccia per stampante che permette di utilizzare una stampante parallela; ciò significa che ogni linea di dati del bus viene collegata ad un corrispondente input della stampante. I dati vengono trasferiti molto più velocemente usando un'interfaccia parallela piuttosto che una interfaccia seriale in quanto quest'ultima deve prima preparare il formato di un byte ed inserire poi informazioni di sincronizzazione.

Interfaccia Seriale
Questo termine si riferisce soprattutto all'interfaccia RS232 ma esistono altri standard per la comunicazione seriale dei dati.

Intero
Un numero senza il punto decimale.

Interprete
La parte del software di sistema che interpreta un linguaggio ad alto livello per fare in modo che possa essere compreso dalla CPU. Ad esempio il codice BASIC immesso da tastiera viene convertito nel linguaggio interno del computer.

Istruzione
Un comando che chiede al computer di eseguire una data operazione. Una sequenza di istruzioni formano un programma.

Iterazione
Uno degli elementi del calcolo. Il computer esegue tutti i compiti spezzandoli in parti più semplici che possono essere gestite dalla CPU. Per fare ciò il computer deve continuare ad eseguire molti elementi più semplici fino al verificarsi di una particolare condizione.

Joystick
Un dispositivo di input che sostituisce generalmente le funzioni dei tasti cursore e rende molti giochi più facili e più semplici.

K
Abbreviazione di Kilo che nel mondo dei calcolatori è molto utilizzato per kilobyte che corrisponde a 1024 byte e cioè a 2 elevato a 10.

Leggibile da una macchina
Un mezzo o una informazione che può essere direttamente compresa da un computer senza immissione dalla tastiera.

Lettura e scrittura
Un attributo assegnato ad un disco, ad un file o ad un drive che permette di scrivervi o leggervi dei dati.

Linguaggi di alto livello
Linguaggi scritti in una forma facilmente comprensibile in cui il vero linguaggio fa la maggior parte del lavoro di interpretazione. Sono più lenti dei programmi in linguaggio macchina ma molto più comprensibili.

Linguaggio a basso livello
Ad esempio il linguaggio Assembler. E' un linguaggio di programmazione in cui ogni istruzione corrisponde ad una istruzione della macchina.

Linguaggio di programmazione
E' ciò che permette di scrivere un programma; è costituito da un insieme di rigide regole sull'uso delle parole e dei numeri.

LISP
Abbreviazione di LISt Processing language o linguaggio per la gestione delle liste. E' un altro linguaggio ad alto livello per computer.

Logica
Componenti elettronici che eseguono le elementari operazioni e funzioni logiche che compongono ogni operazione di un computer.

Logo
E' un linguaggio per computer semplice da imparare ed orientato alla grafica usato frequentemente nelle scuole come aiuto all'insegnamento dell'informatica.

LSI
Large Scale Integration (Integrazione spinta a larga scala). Lo sviluppo dei circuiti integrati ha permesso di inserire molte funzioni in frammenti di silicio sempre più piccoli.

Mappa di memoria
La disposizione della memoria con i vari indirizzi e la locazione di memoria delle varie funzioni, ad esempio lo schermo, il sistema di gestione del nastro, ecc.

Matrice
La disposizione di punti che formano la cella di carattere sullo schermo o sulla testina di stampa di una stampante a matrice. E' anche un termine usato in matematica e in informatica per denotare i vettori.

Memoria
Il computer memorizza molte informazioni e dati disposte in modo logico e accessibili uno ad uno. E' anche conosciuto come RAM, memoria ad accesso diretto, dove le informazioni possono essere memorizzate o lette, o ROM dove le informazioni possono essere lette ma non scritte. I dischi e i nastri sono esempi di memoria esterna.

Menù
Un elenco delle varie opzioni che possono essere eseguite in un determinato ambito e che possono essere selezionate dall'utente.

Microprocessore
Un circuito integrato che si trova nel cuore del computer e che esegue le istruzioni che gli vengono presentate dall'interprete BASIC.

Modem
Un MODulatore/DEMulatore che permette di collegare il computer ad una linea telefonica o a un altro dispositivo di trasmissione seriale comprese le fibre ottiche. Vedere anche accoppiatore acustico.

Modo console
E' il modo diretto di CP/M; sullo schermo appare la A> ed il sistema attende un input di un comando CP/M o di un programma di utilità.

Modo grafico
I primi microcomputer dovevano essere predisposti appositamente per gestire o i caratteri o la grafica. I moderni personal computer possono utilizzare simultaneamente testo e grafica.

Modulatore RF
Un dispositivo che permette di inviare l'output video di un computer alla presa d'antenna di un normale televisore.

Monitor
Lo schermo di un computer; il termine descrive anche il programma in linguaggio macchina che permette di accedere alle operazioni più a basso livello del computer.

Mouse
Una piccola scatola che contiene una sfera di gomma. Spostato su una superficie permette di muovere il cursore a piacimento. E' stato progettato per superare la paura della tastiera e per rendere il software più amichevole.

Nibble
Mezzo byte. Una espressione binaria a quattro bit. Ogni cifra di un numero esadecimale rappresenta un nibble.

Nome di file ambiguo
Un nome di file contenente uno o più caratteri jolly. Tali nomi possono fare riferimento a più di un file.

Notazione decimale
Sistema di numerazione in base 10, nel quale si usano cifre da 0 a 9, che rappresentano unità, decine, centinaia, migliaia, ecc.

Notazione esadecimale

Sono numeri in base sedici. Indicati nel 464/6128 da prefisso & o &H.

Notazione polacca inversa
E' un metodo utilizzato da alcuni calcolatori per eseguire le operazioni aritmetiche. In tale notazione, gli operatori (+, -, *, /) vengono posti di fronte ai valori ai quali vengono applicati.

Numeri a virgola fissa
Un numero rappresentato, utilizzato e memorizzato con un punto decimale in posizione fissata.

Numeri a virgola mobile
Un numero Reale utilizzato e memorizzato con un punto decimale che può trovarsi nella posizione desiderata. Questo metodo è particolarmente utile quando si utilizzano grossi numeri.

Numeri binari

Un numero rappresentato in notazione binaria. Denotati nella programmazione del 464/6128 dal prefisso &X; ad esempio: &X0101 è equivalente (in decimale) a 5.

Numero casuale
Un numero generato dal computer in modo irripetibile ed imprevedibile.

Numero da leggere bit a bit
In tali numeri l'informazione viene estratta considerando lo stato di ognuno degli otto bit che compongono un byte. Il valore decimale intero può non avere significato.

Numero di linee
Il BASIC e alcuni altri linguaggi usano programmi il cui ordine è stabilito dai numeri di linea.

Numero intero
Un numero senza parte frazionale, cioè un numero senza virgola. E' il contrario del numero reale che è composto da una parte intera e una parte frazionale.

Numero reale
Un numero che ha una parte intera ed una frazionale. Una variabile reale rimane tale anche se in un dato momento dell'esecuzione del programma può contenere un numero intero.

Off line
La periferica di un computer (un terminale o una stampante) che non è connessa in modo attivo o utilizzabile dall'unità di elaborazione.

On line
L'opposto di off line.

Operatore
La parte di un'espressione aritmetica che permette di modificare dei numeri, ad esempio +, -, / ecc.

Orologio
Il sistema di temporizzazione di un computer usato per sincronizzare e gestire le operazioni di un computer. L'orologio in tempo reale mantiene l'ora, la data ecc.

Ottale
Un sistema numerico in base otto in cui ogni cifra (da 0 a 7) è composto da tre bit.

Output
Tutto quanto esce da un computer come risultato di qualche calcolo.

Paddle
Nome alternativo per il joystick.

Pagina zero
E' la parte della memoria dell'ambiente CP/M che si trova tra &0000 e &0100 e che viene utilizzata per contenere inportanti parametri di funzionamento.

Parola chiave
Una parola il cui uso all'interno di un programma o di un linguaggio è riservato per una particolare funzione o comando.

Parola riservata
Una parola che ha un significato particolare all'interno di un programma e che non può perciò essere utilizzata per altri scopi. Ad esempio, il BASIC non potrà accettare NEW come nome di variabile in quanto essa è già riservata per altri scopi.

Pascal
Un linguaggio di programmazione strutturato ad alto livello che deve essere compilato prima di essere eseguito e che quindi viene eseguito molto velocemente. E' in genere il linguaggio di programmazione, dopo il BASIC, che uno studente apprende.

Passo della tartaruga
La più piccola distanza che la tartaruga può coprire. Normalmente è un pixel.

PEEK
La funzione BASIC che permette di leggere direttamente la memoria del computer e restituisce il valore di una data locazione.

Penna ottica
Un dispositivo alternativo di input che utilizza una penna.

Periferica
Stampanti, Modem, Joystick, drive per dischi - tutto quanto si collega al computer per espandere le sue capacità.

Pixel
La più piccola parte di schermo che può essere controllata dall'hardware.

Plotter
Un particolare tipo di stampante che permette di tracciare disegni utilizzando delle penne invece di una testina di stampa. E' usato per output tecnico e grafico.

POKE
La parola chiave del BASIC che permette di inserire un dato valore in una specifica locazione di memoria.

Porta
Le porte logiche permettono il passaggio di dati al verificarsi di determinate condizioni. Vi sono diversi tipi di porte (OR, AND, XOR, ecc.). Si veda Algebra booleana.

Porta
Un punto di collegamento per l'input e l'output dei dati.

Portabilità
Indica la possibilità di utilizzare un determinato programma su vari computer - ciò è permesso da sistemi operativi compatibili come CP/M della Digital Research.

Primitive
Procedure, operazioni o comandi che costituiscono il Dr.Logo; le procedure interne.

Programma
E' l'insieme di istruzioni che fa risolvere al computer un determinato problema; può andare da una semplice routine in codice macchina ad un intero programma di elaborazione di testi.

Programma di avviamento
I programmi e sistemi operativi non si caricano da soli ma vengono caricati da una piccola routine situata (di solito) nella ROM che effettua il processo di caricamento in una locazione di memoria specifica.

Programmazione strutturata
Una tecnica di programmazione che permette di produrre programmi che seguono passi ben definiti.

Programmi applicativi
E' un programma che invece di essere uno strumento di applicazione generale ha un compito preciso ad esempio un assemblatore, un driver per stampanti ecc...

Programmi transienti
Una utility di CP/M come ad esempio PIP che può essere caricata nella TPA e può essere eseguita inserendo il nome dalla tastiera.

PROM
Programmable Read Only Memory. Un circuito integrato che una volta scritto non può essere cancellato (vedere anche EPROM).

Prompt
Simbolo o messaggio visualizzato sullo schermo che permette all'utente di rispondere al computer o continuare un input. Il BASIC usa un semplice punto di domanda per richiedere dell'input o la parola Ready quando attende un comando.

Protezione dalla scrittura
Permette di evitare di modificare un file o un disco. un disco protetto da scrittura è un file a sola lettura.

QWERTY
Un termine di uso comune che descrive una tastiera inglese.

RAM
Random Access Memory o memoria ad accesso diretto. E' un tipo di memoria che può essere letto o scritto, mediante i circuiti del computer in fase di esecuzione del programma

Raster
Un tipo di scrittura sullo schermo, in cui le immagini sono composte da un certo numero di scansioni orizzontali.

Refresh
Permette di aggiornare le informazioni, sullo schermo o in memoria. Non è necessariamente un processo distruttivo, a volte ripristina semplicemente ciò che è già presente sullo schermo o in memoria.

Registro
Una locazione di memoria all'interno della CPU.

REM
Una istruzione che non viene eseguita ma serve a ricordare al programmatore che cosa il programma sta facendo e quando il programma è stato fatto.

Rete
Quando due computer o più sono collegati insieme per scambiare dati e informazioni o via cavo o via modem.

Riconoscimento ottico dei caratteri
Un mezzo per leggere caratteri stampati o scritti utilizzando un lettore ottico.

Riconoscimento vocale
La conversione delle parole in istruzioni leggibili da una macchina.

Ricorsione
Un insieme di passi all'interno di un programma o di una routine in cui il risultato di ogni ciclo è in relazione con il precedente.

Risoluzione
La capacità di stabilire dove termina un elemento dello schermo e dove inizia il successivo. Indica anche la possibilità del computer di eseguire operazioni su numeri molto grossi.

ROM
Read Only Memory, memoria a sola lettura, una volta scritta non può essere né cancellata, né riscritta.

Routine
Parte del programma che esegue un compito ben determinato. Può trovarsi all'interno del programma o al suo esterno, in modo che possa essere incorporata. Un programma che genera un orologio può essere considerato una routine.

RS232C
Uno standard per le comunicazioni seriali. Entrambi i dispositivi che si trovano alle estremità devono essere configurati secondo le caratteristiche della trasmissione. Si confronti con l'interfaccia Centronics.

Rumore

Le possibilità sonore del 464/6128 permettono di inserire una quantità variabile di rumore usando il comando sound in modo da creare effetti come ad esempio delle esplosioni.

Scorrimento
E' il modo in cui lo schermo scorre quando viene raggiunto il fondo.

Separatore
Esegue la stessa funzione di un delimitatore: indica i confini tra le parole riservate ed altri elementi di un programma o dei dati.

Set di caratteri
Tutte le lettere, numeri e simboli disponibili su un computer o su una stampante. Il fatto che un carattere esista su un computer non implica necessariamente che questo sia disponibile su un stampante.

Settore
Un blocco di dati su disco. Il sistema AMSTRAD usa un settore di 512 bytes.

Simulazione
Tecnica per sperimentare su computer alcuni processi della vita reale, ad esempio il volo, la guida di un'auto, ecc.

Sintesi vocale
Generazione di parole usando hardware e software

Sistema operativo
Il sorvegliante del "parcheggio" costituito dalla memoria del computer. E' composto da software che gestisce le precedenze e le sincronizzazioni nelle operazioni del computer.

Software
Vedere Programmi.

Sola lettura
Un attributo assegnato ad un disco, ad un file o a un drive che evita di scrivere su di esso dei dati.

Sovrascrittura
Cancellazione di un'area di memoria e sostituzione del suo contenuto con altri dati.

Sprite
Un carattere dello schermo che si può muovere liberamente per lo schermo.

Stack
Un'area di memoria che serve ad "impilare" informazioni, ma in cui solo l'ultimo elemento della fila può essere richiamato.

Stampante
Dispositivo che permette di stampare testi e grafica.

Stampante a margherita
Una stampante che può produrre documenti di alta qualità. I caratteri stampati vengono creati mediante l'impatto delle lettere contro il nastro inchiostrato.

Stringa di caratteri
Una parte di dato <variabile> comprendente una sequenza di caratteri che può essere memorizzata o trattata come una singola unità, cioè una parola o un insieme di parole.

Subroutine
Vedere Routine.

Tabelle di verità
Il risultato di un'operazione logica può essere vero o valso, 1 o 0. Le tabelle di verità indicano i valori di verità del risultato di un'operazione logica.

Tartaruga
Un simbolo grafico a forma di punta di freccia che viene utilizzato come cursore grafico durante l'utilizzo di Dr.Logo.

Tasti definibili dall'utente

Il 464/6128 ha 32 tasti che possono essere ridefiniti per eseguire compiti più complessi.

Tasti di controllo del cursore
Tasti che spostano il cursore nello schermo e che sono frequentemente usati per controllare la direzione di azione nei giochi; sono indicati da frecce.

Tasti funzione

Ad essi è stato assegnato un compito specifico che si può aggiungere o sostituire a quanto è scritto sopra il tasto. Alcuni tasti del 464/6128 possono essere definiti come tasti funzione, nel qual caso la pressione di un singolo tasto può provocare l'esecuzione di più istruzioni anche piuttosto complesse.

Tastiera
L'insieme dei tasti alfanumerici di cui il computer è dotato per permettere l'inserimento di comandi e altre informazioni.

Tastierino numerico

L'area sulla tastiera dove si trovano i tasti numerici in modo da facilitare l'inserimento dei dati numerici. Nel caso del 464/6128 vi si trovano anche i tasti funzione definibili dall'utente.

Tavoletta grafica
Un dispositivo che permette di prelevare le coordinate dei punti di un certo disegno per manipolarlo all'interno del computer. E' una forma di conversione analogica digitale.

Tecnologia dell'informazione
Tutto ciò che è in relazione con l'uso dell'elettronica nell'elaborazione dell'informazione e nelle comunicazioni: l'elaborazione di testi, le comunicazioni di dati, Videotel ecc.

Terminale
Una tastiera con uno schermo o una stampante che permette di collegarsi ad un computer.

Terminale generico
Un terminale per computer che si comporta semplicemente come un dispositivo di input ed output e non elabora in alcun modo le informazioni. Un terminale stupido è invece un terminale in cui non esiste nessun circuito per la gestione del video ed in cui le informazioni vengono ricevute semplicemente come un segnale video.

TPA - Transient program area
L'area di memoria che va da &0100 dove i programmi CP/M e i programmi utente vengono eseguiti e memorizzano dati.

Tracce di sistema
Tracce del disco riservate dal sistema CP/M.

Traccia
Sono anelli concentrici su un disco. Ogni traccia contiene un numero fisso di settori. Le tracce ed i settori vengono fissati durante la formattazione.

Troncamento
Un numero o stringa a cui è stata tolta la testa o la coda.

Utility
Un programma che esegue una operazione specifica, ad esempio l'ordinamento o la copia di un file.

Variabile
Un oggetto contenuto in un programma che può essere richiamato mediante il nome ma il cui valore può cambiare durante l'esecuzione del programma.

Vettore
Una matrice bidimensionale in cui i dati vengono memorizzati secondo coordinate orizzontali e verticali.

XYZY
Una parola magica che permette di uscire da brutte situazioni in un gioco di avventura.

Appendice 3
Alcuni programmi ...

Bustout

Così semplice ed avvincente! Un giocatore contro il computer. Tastiera o Joystick.

```
10 'BUSTOUT by ALEXANDER MARTIN
20 'copyright (c) AMSOFT 1984
30 '
40 MODE 1:BORDER 1:INK 0,1:INK 1,26:INK 2,24:INK 3,6
50 SPEED KEY 15,2
60 ENV 1,1,18,0,11,0,10
70 ENT 1,10,2,2
80 ENV 3,1,0,16,5,-3,2
90 ENV 2,5,3,3,1,-21,22,9,-3,2
100 ENT -2,10,2,2,5,-7,1,2,11,3,2,-4,8
110 '
120 '
130 MOVE 30,32:DRAWR 0,400,1:MOVE 610,32:DRAWR 0,400,1
140 PEN 3:LOCATE 3,1:PRINT STRING$(36,143)
150 PEN 2:LOCATE 3,2:PRINT STRING$(36,143)
160 PEN 1:FOR r=5 TO 6:LOCATE 3,r:PRINT STRING$(36,143)
    :NEXT r
170 bx=9
180 lives=5:score=0
190 PEN 1:GOSUB 680:CLEAR INPUT
200 IF INKEY$<>CHR$(32) AND JOY(0)<16 THEN 200
210 LOCATE 11,23:PRINT SPACE$(20):LOCATE 1,24:PRINT SPA
    CE$(40);
220 GOSUB 690:GOSUB 660:GOTO 280
230 '
240 '
250 LOCATE bx,24:PRINT"  ";STRING$(4,131);"  ":RETURN
260 '
270 '
280 xa=1:ya=1:IF INT(RND*2)=1 THEN xa=-xa
290 PEN 1:GOSUB 250
300 ORIGIN 0,400
310 x=bx+4:y=11:x1=x:y1=y
320 '
330 '
340 x1=x+xa:y1=y+ya
```

continua sulla prossima pagina

```
350 IF x1=3 OR x1=38 THEN xa=-xa
360 GOSUB 540
370 IF y1=24 AND x1>bx+1 AND x1<bx+6 THEN ya=-ya:y1=y1-
    2:SOUND 130,44,8,7,1,1:a=((x>bx+5)OR(x<bx+2)):IF a=
    -1 THEN xa=xa*a:x1=x1+xa:y1=y1+1
380 IF y1=25 THEN LOCATE x,y:PRINT" ":GOTO 500
390 GOSUB 250
400 t=TEST((16*x1)-1,-(16*y1)-1)
410 IF t<>0 THEN ya=-ya:xz=x1:yz=y1:y1=y1+ya:GOSUB 590:
    IF t=2 THEN score=score+10:GOSUB 660
420 IF t=3 THEN score=score+20:GOSUB 660
430 IF t=1 THEN score=score+5:GOSUB 660
440 IF y1=1 THEN ya=1
450 LOCATE x,y:PRINT " ":LOCATE x1,y1:PRINT CHR$(233):x
    =x1:y=y1
460 IF y=1 OR x=3 OR x=38 THEN SOUND 129,78,8,7,1,1
470 GOTO 340
480 '
490 '
500 lives=lives-1:SOUND 132,19,46,12,2,2:IF lives=0 THE
    N GOTO 620
510 GOSUB 660:GOTO 280
520 '
530 '
540 IF (INKEY(8)=0 OR INKEY(74)=0) AND bx>2 THEN bx=bx-
    2:RETURN
550 IF (INKEY(1)=0 OR INKEY(75)=0) AND bx<32 THEN bx=bx
    +2:RETURN
560 RETURN
570 '
580 '
590 LOCATE xz,yz:PRINT " ":RETURN
600 '
610 '
620 IF score>=hiscore THEN hiscore=score
630 GOSUB 660:score=0:lives=5:GOTO 130
640 '
650 '
660 SOUND 130,0,20,13,3,0,31:LOCATE 1,25:PRINT TAB(4)"H
    ISCORE";hiscore;
670 LOCATE 18,25:PRINT"SCORE";score:LOCATE 30,25:PRINT"
    LIVES";lives:RETURN
680 LOCATE 11,23:PRINT"PRESS SPACE TO START":RETURN
690 LOCATE 1,25:PRINT SPACE$(40);:RETURN
```

Bomber

Una variazione di un tema classico! Un giocatore contro il computer. Solo tastiera.

```
10 'BOMBER by DAVE TOWN
20 'copyright (c) AMSOFT 1984
30 '
40 MODE 1:CLS:INK 0,0:BORDER 0:INK 1,18:INK 2,6:INK 3,4
   :INK 5,15:INK 6,2:INK 7,24:INK 8,8:INK 9,26:INK 10,1
   0:INK 11,20:INK 12,12:INK 13,16:INK 14,14:INK 15,21
50 SYMBOL AFTER 240:SYMBOL 241,&40,&60,&70,&7F,&7F,&3F,
   &7,&0:SYMBOL 242,&0,&32,&7A,&FE,&FA,&F2,&E0,&0
60 score=0:hiscore=0:plane$=CHR$(241)+CHR$(242):x=2:y=2
   :drop=0:a=2:b=2
70 GOSUB 480
80 CLS
90 PEN 2:LOCATE 1,15:INPUT"Enter skill : 0 (ACE) to 5 (
   NOVICE) : ",skill
100 IF skill<0 OR skill>5 GOTO 90
110 skill=skill+10
120 LOCATE 1,15:PRINT CHR$(18);:LOCATE 1,15:INPUT"Enter
     speed 0 (FAST) to 100 (SLOW) : ",rate
130 IF rate>100 OR rate<0 GOTO 120
140 '
150 'Buildings
160 '
170 MODE 0:FOR base=5 TO 15:FOR height=21 TO INT(RND(1)
    *8+skill) STEP -1:LOCATE base,height:PEN base-2:PRI
    NT CHR$(143)+CHR$(8)+CHR$(11)+CHR$(244);:NEXT :NEXT
180 PLOT 0,20,4:DRAW 640,20,4
190 LOCATE 1,25:PEN 2:PRINT"SCORE";score;:LOCATE 13,25:
    PRINT"HI";hiscore;
200 '
210 'Main Game
220 '
230 LOCATE x-1,y:PRINT"   ";
240 PEN 1:LOCATE x,y:PRINT plane$;:PEN 2
250 IF y=21 AND x=15 THEN GOTO 290:ELSE GOTO 340
260 '
270 'Landed
280 '
290 FOR c=0 TO 1000:NEXT
```

continua sulla prossima pagina

```
300 score=score+100-(skill*2):skill=skill-1:x=2:y=2:a=2
    :b=2:drop=0
310 IF skill<10 THEN skill=10:rate=rate-20
320 IF rate<0 THEN rate=0
330 GOTO 150
340 FOR c=0 TO rate:NEXT
350 x=x+1
360 IF x=18 THEN LOCATE x-1,y:PRINT CHR$(18);:x=2:y=y+1
    :LOCATE x,y:PEN 1:PRINT plane$;:PEN 2
370 a$=INKEY$:IF a$=" " AND drop=0 THEN drop=1:b=y+2:a=
    x
380 IF y=21 THEN drop=0
390 IF drop=1 THEN LOCATE a,b:PRINT CHR$(252);:LOCATE a
    ,b-1:PRINT" ";:b=b+1:IF b>21 THEN LOCATE a,b:PRINT"
     ";:LOCATE a,b-1:PRINT" ";:a=0:b=0:drop=0:SOUND 3,4
    000,10,12,0,0,10
400 ga=(a-0.5)*32:gb=400-(b*16):bomb=TEST(ga,gb)
410 IF bomb>0 THEN GOTO 670
420 gx=((x+1.5)*32):gy=408-(y*16):crash=TEST(gx,gy)
430 IF crash>0 GOTO 570
440 GOTO 230
450 '
460 'Instructions
470 '
480 LOCATE 1,2:PEN 1:PRINT"You are piloting an aircraft
     over a des-erted city and must clear the buildings
     in order to land and refuel.  Your air- craft move
    s across  the screen from left to right.";:PRINT
490 PRINT:PRINT"On reaching the right, the aircraft
     returns to the left A LINE FURTHER DOWN.You have a
    n unlimited supply of bombs   and you can drop them
     on the buildings  below by pressing the SPACE BAR.
    ";:PRINT
500 PRINT:PRINT"Each time you land, the height of the
     buildings or the speed of your aircraft increases.
    ";:PRINT:PRINT:PRINT"ONCE YOU HAVE RELEASED A BOMB,
     YOU WILL NOT BE ABLE TO RELEASE ANOTHER UNTIL THE
    FIRST HAS EXPLODED !!!!";
510 PEN 2:LOCATE 1,24:PRINT:PRINT"Press any key to star
    t.";
520 a$=INKEY$:IF a$="" GOTO 520
530 RETURN
540 '
550 'Collision
```

continua sulla prossima pagina

```
560 '
570 LOCATE x-1,y:PRINT CHR$(32)+CHR$(32)+CHR$(32)+CHR$(
    253)+CHR$(8)+CHR$(238)+CHR$(8);
580 FOR t=1 TO 10:SOUND 7,4000,5,15,0,0,5:PEN t:PRINT C
    HR$(253)+CHR$(8)+CHR$(238)+CHR$(8)+CHR$(32)+CHR$(8)
    ;:FOR tm=0 TO 50:NEXT:NEXT:PEN 2
590 CLS:LOCATE 1,5:PRINT"You scored";score;
600 IF score>hiscore THEN hiscore=score:LOCATE 1,8:PRIN
    T"TOP SCORE!!";
610 score=0:LOCATE 1,12:PRINT"Press R to restart";
620 a$=INKEY$:IF a$="r" OR a$="R" GOTO 630 ELSE GOTO 62
    0
630 PEN 1:MODE 1:x=2:y=2:a=2:b=2:GOTO 90
640 '
650 'Bombed building
660 '
670 LOCATE a,b-1:PRINT" "+CHR$(8);:PEN 4:FOR tr=1 TO IN
    T(RND(1)*3)+1:score=score+5:SOUND 3,4000,10,12,0,0,
    10:LOCATE a,b:FOR t=0 TO 4:PRINT CHR$(253)+CHR$(8)+
    CHR$(32)+CHR$(8);:NEXT:b=b+1
680 IF b=24 THEN b=b-1
690 NEXT
700 LOCATE 6,25:PRINT score;:drop=0:a=x:b=y:GOTO 230
```

Telly tennis

Semplice ma di gran divertimento! Per due giocatori o un giocatore contro il computer. Tastiera o 1 o 2 Joystick.

```
10 'TELLY TENNIS by DAVID RADISIC
20 'copyright (c) AMSOFT 1985
30 '
40 DEFINT a-z
50 comp=1
60 ENV 1,=11,20,=9,5000
70 MODE 1:INK 0,10:BORDER 10:INK 1,26:INK 2,18:INK 3,0
80 GOSUB 710
90 GOSUB 150
100 GOSUB 330
110 GOSUB 420
120 LOCATE 13,1:PRINT USING"#### ";score1;
130 LOCATE 35,1:PRINT USING"#### ";score2;
140 GOTO 100
150 PEN 2
160 x(1)=3:y(1)=5
170 x(2)=37:y(2)=22
180 edge$=CHR$(233):edge2$=STRING$(2,207)
190 LOCATE 1,3:PRINT STRING$(39,edge$):PRINT STRING$(39
    ,edge$)
200 FOR i=1 TO 19
210 PRINT edge2$;TAB(38);edge2$
220 NEXT
230 PRINT STRING$(39,edge$):PRINT STRING$(39,edge$);
240 WINDOW #1,3,37,5,23
250 CLS#1
260 SYMBOL 240,0,60,126,126,126,126,60,0
270 bat$="|"+CHR$(8)+CHR$(10)+"|"
280 clr$=" "+CHR$(8)+CHR$(10)+" "
290 ball$=CHR$(240)
300 PEN 3
310 LOCATE 2,1:PRINT"Player 1 :    0";:LOCATE 24,1:PRIN
    T"Player 2 :    0";
320 RETURN
330 n=INT(RND*2):CLS #1:scored=0
340 PEN 3
350 FOR i=1 TO 2:LOCATE x(i),y(i):PRINT bat$;:NEXT
360 ON n GOTO 390
```

continua sulla prossima pagina

```
370 xb=21:dx=1
380 GOTO 400
390 xb=19:dx=-1
400 yb=12:dy=INT(RND*3)-1
410 RETURN
420 GOSUB 600
430 oxb=xb:oyb=yb
440 GOSUB 500
450 IF note>0 THEN SOUND 129,note,50,15,1
460 LOCATE oxb,oyb:PRINT" ";
470 LOCATE xb,yb:PRINT ball$
480 IF scored=0 THEN 420
490 RETURN
500 LOCATE xb+dx,yb+dy:ch$=COPYCHR$(#0)
510 note=0
520 IF ch$=" " THEN xb=xb+dx:yb=yb+dy:RETURN
530 IF ch$="|" THEN dx=2-dx-2:dy=INT(RND*3)-1:note=200:
    RETURN
540 IF ch$=LEFT$(edge2$,1) THEN 570
550 IF ch$=edge$ THEN dy=2-dy-2:note=250
560 RETURN
570 IF dx>0 THEN score1=score1+1 ELSE score2=score2+1
580 scored=1:note=2000
590 RETURN
600 p(1)=(INKEY(69)>=0)+(INKEY(72)>=0)+ABS((INKEY(71)>=
    0)+(INKEY(73)>=0))*2
610 IF comp=1 THEN p(2)=ABS(y(2)<yb)*2+(y(2)>yb):GOTO 6
    30
620 p(2)=(INKEY(4)>=0)+(INKEY(48)>=0)+ABS((INKEY(5)>=0)
    +(INKEY(49)>=0))*2
630 PEN 3
640 FOR i=1 TO 2
650 LOCATE x(i),y(i)+p(i):ch$=COPYCHR$(#0)
660 IF ch$=" " THEN LOCATE x(i),y(i):PRINT clr$;:y(i)=y
    (i)+ROUND(p(i)/2)
670 LOCATE x(i),y(i):PRINT bat$;
680 NEXT
690 PEN 1
700 RETURN
710 PEN 2:PRINT:PRINT TAB(15)"Ping-Pong":PRINT TAB(15)"
    ---------"
720 PEN 3:PRINT:PRINT TAB(14)"To move bats :
730 PRINT:PRINT:PEN 1
740 PRINT" Player 1          Player 2          Direction":PRI
    NT
```

continua sulla prossima pagina

```
750 PRINT"    A                6                  UP"
760 PRINT"    Z                3                  DOWN":PRINT
770 PEN 3:PRINT:PRINT TAB(14)"Or joysticks"
780 PRINT:PRINT:PRINT:PRINT
790 PEN 2
800 PRINT TAB(6)"Select <1> or <2> players"
810 i$=INKEY$:IF i$<>"1" AND i$<>"2" THEN 810
820 IF i$="1" THEN comp=1 ELSE comp=0
830 MODE 1:RETURN
```

Electric fencing

Si provi a battere l'avversario! Solo per due giocatori. Tastiera o Joystick.

```
10 'ELECTRIC FENCING by ALEXANDER MARTIN
20 'copyright (c) AMSOFT 1985
30 '
40 DEFINT a-z
50 MODE 0
60 GOSUB 980
70 GOSUB 1370
80 GOSUB 270
90 GOSUB 1520
100 GOSUB 1370
110 GOSUB 1270
120 '
130 '
140 REM start
150 IF finished THEN GOTO 100
160 GOSUB 240
170 FRAME:IF p1dir THEN GOSUB 570 ELSE FRAME:FRAME
180 FRAME:IF p2dir THEN GOSUB 620 ELSE FRAME:FRAME
190 IF p1sa=-1 THEN GOSUB 670
200 IF p2sa=-1 THEN GOSUB 720
210 GOTO 140
220 '
230 '
240 IF j THEN  380 ELSE  480
250 '
260 '
270 CLS:PEN 6
280 PRINT:PRINT"   CHOOSE CONTROL"
290 PRINT:PRINT:PRINT:PRINT"press J/K then ENTER"
300 LOCATE 4,10:PRINT"JOYSTICK";TAB(5);"OR KEYS"
310 LOCATE 12,10:IF j THEN PRINT"*":ELSE PRINT" "
320 LOCATE 12,11:IF j THEN PRINT" ":ELSE PRINT"*"
330 IF NOT(INKEY(45)) THEN j=-1
340 IF NOT(INKEY(37)) THEN j=0
350 IF NOT(INKEY(18)) THEN RETURN ELSE 310
360 '
370 '
380 p1=JOY(0):p2=JOY(1)
390 p1dir=(p1 AND 1)*-1+(p1 AND 2)*0.5
```

continua sulla prossima pagina

```
400 p2dir=(p2 AND 1)*-1+(p2 AND 2)*0.5
410 IF P1 AND 16 THEN p1sa=p1sa-1:IF p1sa=-1 THEN AFTER
     15 GOSUB 770
420 IF P2 AND 16 THEN p2sa=p2sa-1:IF p2sa=-1 THEN AFTER
     15 GOSUB 770
430 IF p1sa THEN p1dir=0
440 IF p2sa THEN p2dir=0
450 RETURN
460 '
470 '
480 p2dir=((INKEY(4)=0)*1)+((INKEY(5)=0)*-1)
490 p1dir=((INKEY(69)=0)*1)+((INKEY(71)=0)*-1)
500 IF INKEY(63)=0 THEN p1sa=p1sa-1:IF p1sa=-1 THEN AFT
    ER 15 GOSUB 770
510 IF INKEY(10)=0 THEN p2sa=p2sa-1:IF p2sa=-1 THEN AFT
    ER 15 GOSUB 770
520 IF p1sa THEN p1dir=0
530 IF p2sa THEN p2dir=0
540 RETURN
550 '
560 '
570 pt=p1wp+p1dir:IF pt>25 OR pt<6 THEN RETURN ELSE p1w
    p=pt
580 p1dir=0
590 PEN 1:LOCATE 3,p1wp:CLS #3:PRINT CHR$(209);:RETURN
600 '
610 '
620 pt=p2wp+p2dir:IF pt>25 OR pt<6 THEN RETURN ELSE p2w
    p=pt
630 p2dir=0
640 PEN 2:LOCATE 18,p2wp:CLS #5:PRINT CHR$(211);:RETURN
650 '
660 '
670 PAPER #4,4:WINDOW #4,4,17,p1wp,p1wp:CLS#4:FRAME:FRA
    ME
680 PAPER #4,0:CLS#4
690 GOTO 570
700 '
710 '
720 PAPER #6,5:WINDOW #6,4,17,p2wp,p2wp:CLS#6:FRAME:FRA
    ME
730 PAPER #6,0:CLS#6
740 GOTO 620
750 '
```

continua sulla prossima pagina

```
760 '
770 pwpe=(p1wp=p2wp):IF p1sa AND NOT(p2sa) AND pwpe THE
    N p1sc=p1sc+1:SOUND 132,120,10,0,1,0:PRINT#1,a$(p1s
    c);:IF p1sc=9 THEN 860
780 IF p2sa AND NOT(p1sa) AND pwpe THEN p2sc=p2sc+1:SOU
    ND 132,100,10,0,1,0:PRINT#2,a$(p2sc);:IF p2sc=9 THE
    N 860
790 IF p1sa THEN SOUND 132,40,70,0,1,1
800 IF p2sa THEN SOUND 132,56,70,0,1,1
810 p1sa=0
820 p2sa=0
830 RETURN
840 '
850 '
860 PEN 6
870 LOCATE 6,10:PRINT"GAME OVER"
880 IF p1sc=9 THEN INK 1,2,20:INK 2,0 ELSE INK 2,6,17:I
    NK 1,0
890 SOUND 129,1000,0,12,3:SOUND 130,900,0,12,3
900 WHILE INKEY$<>"":WEND
910 t!=TIME:WHILE t!+2000>TIME:WEND
920 WHILE INKEY$="":WEND
930 CLS
940 finished=-1
950 RETURN
960 '
970 '
980 a$(0)="111101101101111"
990 a$(1)="001001001001001"
1000 a$(2)="111001111100111"
1010 a$(3)="111001111001111"
1020 a$(4)="100100101111001"
1030 a$(5)="111100111001111"
1040 a$(6)="111100111101111"
1050 a$(7)="111001001010010"
1060 a$(8)="111101111101111"
1070 a$(9)="111101111001001"
1080 FOR n=0 TO 9
1090 howlong=LEN(a$(n))
1100 FOR n2=1 TO howlong
1110 IF MID$(a$(n),n2,1)="1"THEN MID$(a$(n),n2,1)=CHR$(
     143)ELSE MID$(a$(n),n2,1)=CHR$(32)
1120 NEXT n2,n
1130 '
```

continua sulla prossima pagina

```
1140 '
1150 b$="ELECTRIC FENCING"
1160 c$=CHR$(32)+CHR$(164)+" Alexander Martin"
1170 ENV 1,=9,2000:ENT -1,6,3,1
1180 ENV 2,127,0,0,127,0,0,127,0,0,127,0,0,127,0,0
1190 ENV 3,=9,9000
1200 '
1210 '
1220 BORDER 0
1230 INK 0,12:PEN #4,1:PEN #6,2:PEN #1,1:PEN #2,2:PAPER
      #1,3:PAPER #2,3:PEN #0,6
1240 RETURN 'FROM SETTING UP CONSTANTS
1250 '
1260 '
1270 INK 0,12:INK 1,2:INK 2,6:INK 3,13:INK 4,20:INK 5,1
     7:INK 6,20
1280 WINDOW #3,3,3,6,25:WINDOW #5,18,18,6,25
1290 WINDOW #1,3,5,1,5:WINDOW #2,16,18,1,5:WINDOW #7,1,
     20,1,5:PAPER #7,3
1300 CLS:CLS#7:PRINT#1,a$(0);:PRINT#2,a$(0);:p1sc=0:p2s
     c=0:p1wp=5:p2wp=24:p1dir=1:p2dir=1
1310 GOSUB 570:GOSUB 620
1320 SOUND 1,1000,0,12,2:SOUND 2,900,0,12,2
1330 p1sa=0:p2sa=0:finished=0
1340 RETURN 'FROM GAME SHEET RESTORE
1350 '
1360 '
1370 CLS
1380 PEN 7
1390 FOR n=1 TO LEN(b$)
1400 LOCATE 2+n,10
1410 FOR n2=LEN(b$) TO n STEP-1
1420 PRINT MID$(b$,n2,1)
1430 LOCATE 2+n,10
1440 SOUND 135,20*n2,5,12,2,1
1450 NEXT n2,n
1460 SOUND 135,100,0,13,3,1,20
1470 PEN 6:PRINT:PRINT:PRINT:PRINT c$
1480 FOR n=1 TO 5000:NEXT
1490 RETURN
1500 '
1510 '
1520 IF j THEN RETURN
1530 CLS
```

continua sulla prossima pagina

```
1540 LOCATE 7,5
1550 PRINT"CONTROLS"
1560 PRINT
1570 PRINT"  PLAYER1  PLAYER2"
1580 PRINT
1590 PRINT"  A      up      6"
1600 PRINT"  Z     down     3"
1610 PRINT"  X     fire     7"
1620 t!=TIME:WHILE t!+1000>TIME:WEND
1630 RETURN
```

Amthello

Il gioco delle persone che vogliono riflettere. Bisogna cercare di circondare e catturare l'avversario senza lasciarsi catturare! Per un giocatore contro il computer. Solo tastiera.

```
10 'AMTHELLO by M.J.GRIBBINS
20 'copyright (c) AMSOFT 1984
30 '
40 BORDER 14
50 CLEAR
60 MODE 1:PEN 0:PAPER 1:CLS
70 INK 0,0:INK 1,14:INK 2,18:INK 3,26
80 LOCATE 2,3:PEN 3:PRINT"A":LOCATE 3,4:PRINT"M":LOCATE
    4,5:PRINT"T":LOCATE 5,6:PRINT"H"
90 LOCATE 6,7:PRINT"E":LOCATE 7,8:PRINT"L":LOCATE 8,9:P
   RINT"L":LOCATE 9,10:PRINT"O"
100 WINDOW #1,2,39,22,25:PAPER #1,1:PEN #1,0:CLS #1
110 PEN 0
120 LOCATE #1,8,1:PRINT #1,"BLACK ALWAYS PLAYS FIRST"
130 LOCATE #1,1,3:PRINT #1,"PRESS B OR W TO CHOOSE BLAC
    K OR WHITE"
140 B$=INKEY$:IF B$="" THEN 140
150 IF B$="W" OR  B$="w" THEN Q%=3:N%=0:GOTO 210
160 IF B$="B" OR  B$="b" THEN Q%=0:N%=3:GOTO 210
170 CLS #1:LOCATE #1,4,3
180 PRINT #1,"        BLACK OR WHITE ONLY"
190 FOR T=0 TO 1000:NEXT T
200 GOTO 140
210 DIM C%(10,10),P%(9,9),C1%(8),C2%(8),CX%(9),CY%(9)
220 I1%=2:J1%=2:I2%=7:J2%=7
230 FOR I%=0 TO 9
240 C%(I%,0%)=6:C%(0,I%)=6
250 C%(9,I%)=6:C%(I%,9)=6
260 NEXT I%
270 FOR I%=1 TO 8
280 READ C1%(I%),C2%(I%)
290 FOR J%= 1 TO 8
300 READ P%(I%,J%)
310 C%(I%,J%)=6
320 NEXT J%:NEXT I%
330 C%(4,4)=3:C%(4,5)=0:C%(5,4)=0:C%(5,5)=3
```

continua sulla prossima pagina

```
340 FOR K%=1 TO 58
350 READ AR%,BR%,CR%,DR%
360 PLOT AR%,BR%:DRAW CR%,DR%,0
370 NEXT K%
380 GOSUB 1460
390 IF Q%=3 GOTO 770
400 CLS #1:INPUT #1,"  WHICH LINE DO YOU WANT ";E%
410 IF E% <1 OR E% >8 GOTO 400
420 LOCATE #1,1,3:INPUT #1,"WHICH COLUMN DO YOU WANT ";
    D%
430 IF D% <1 OR D% >8 GOTO 420
440 IF C%(D%,E%)=6 GOTO 480
450 CLS #1:LOCATE #1,5,2:PRINT #1,"THAT SQUARE IS ALREA
    DY OCCUPIED !"
460 FOR T=1 TO 1000:NEXT T
470 GOTO 400
480 PLOT 270+(30*D%),70+(30*E%):DRAW 290+(30*D%),89+(30
    *E%),Q%
490 PLOT 290+(30*D%),70+(30*E%):DRAW 270+(30*D%),89+(30
    *E%),Q%
500 GOTO 540
510 FOR M%= 0 TO 19 STEP 2:PLOT 270+(30*D%),70+M%+(30*E
    %)
520 DRAW 290+(30*D%),70+M%+(30*E%),6:NEXT M%
530 GOTO 400
540 VRX%=0
550 FOR K%=1 TO 8
560 VR%=0:C3%=D%:C4%=E%
570 C3%=C3%+C1%(K%):C4%=C4%+C2%(K%)
580 IF C%(C3%,C4%)=N% GOTO 590 ELSE 600
590 VR%=VR%+1:GOTO 570
600 IF C%(C3%,C4%)=6 GOTO 610 ELSE 620
610 NEXT K%:GOTO 670
620 IF VR%=0 GOTO 610 ELSE 630
630 VRX%=VRX%+VR%
640 C3%=C3%-C1%(K%):C4%=C4%-C2%(K%)
650 IF C%(C3%,C4%)=6 GOTO 610 ELSE 660
660 C%(C3%,C4%)=Q%:GOTO 640
670 IF VRX%=0 GOTO 680 ELSE 710
680 CLS #1:PRINT #1," THIS IS NOT A POSSIBLE CHOICE"
690 FOR T=1 TO 1000:NEXT T
700 GOTO 510
710 E%=E%:D%=D%:VRX%=VRX%
720 CLS #1:PRINT #1,"YOU HAVE PLAYED LINE NUMBER ";E%
```

continua sulla prossima pagina

```
730 PRINT #1,"            AND COLUMN NUMBER ";D%
740 LOCATE #1,2,4:PRINT #1,"THAT GIVES YOU ";VRX%;" SQU
    ARES(S)"
750 C%(D%,E%)=Q%:GOSUB 1710
760 GOSUB 1460
770 CLS #1:LOCATE #1,10,2:PRINT #1,"NOW IT'S MY TURN ..
    .!"
780 P%=0:VRX%=0:VRY%=0
790 IF I1%*J1%=1 AND I2%*J2%=64 GOTO 860
800 FOR K%=2 TO 7
810 IF C%(2,K%) <> 6 THEN I1%=1
820 IF C%(7,K%) <> 6 THEN I2%=8
830 IF C%(K%,2) <> 6 THEN J1%=1
840 IF C%(K%,7) <> 6 THEN J2%=8
850 NEXT K%
860 FOR I%=I1% TO I2%
870 FOR J%=J1% TO J2%
880 IF C%(I%,J%)=6 GOTO 1030
890 NEXT J%:NEXT I%
900 IF P% > 0 GOTO 1000
910 IF PAS%=1 GOTO 920 ELSE 940
920 CLS #1:PRINT #1," DEADLOCK ! I MUST PASS ALSO.GAME
    OVER"
930 FOR T=1 TO 1000:NEXT T:GOTO 1550
940 CLS #1:LOCATE #1,18,2:PRINT #1,"I MUST PASS"
950 GOSUB 2720
960 IF PAS%=1 GOTO 970 ELSE 990
970 CLS #1:PRINT #1,"DEADLOCK! YOU MUST PASS ALSO.GAME
    OVER"
980 FOR T=1 TO 1000:NEXT T:GOTO 1550
990 GOTO 400
1000 IF LC%=0 THEN LC%=1:RANDOMIZE LC%:RL%=RND(LC%)
1010 CX1%=CX%(RL%):CX2%=CY%(RL%)
1020 GOTO 1220
1030 VRX%=0
1040 FOR K%=1 TO 8
1050 VR%=0:C3%=I%:C4%=J%
1060 C3%=C3%+C1%(K%):C4%=C4%+C2%(K%)
1070 IF C%(C3%,C4%)=Q% GOTO 1080 ELSE 1090
1080 VR%=VR%+1:GOTO 1060
1090 IF C%(C3%,C4%)=6 GOTO 1100 ELSE 1110
1100 NEXT K%:GOTO 1130
1110 IF VR%=0% GOTO 1100 ELSE 1120
1120 VRX%=VRX%+VR%:GOTO 1100
```

continua sulla prossima pagina

```
1130 IF VRX%=0 GOTO 890
1140 IF P%(I%,J%) < P% GOTO 890
1150 IF P%(I%,J%) > P% GOTO 1160 ELSE 1170
1160 P%=P%(I%,J%):VRY%=VRX%:LC%=0:CX%(0)=I%:CY%(0)=J%:G
     OTO 890
1170 IF VRY% > VRX% GOTO 890
1180 IF VRY% < VRX% GOTO 1190 ELSE 1200
1190 LC%=0:VRY%=VRX%:CX%(0)=I%:CY%(0)=J%:GOTO 890
1200 LC%=LC%+1:CX%(LC%)=I%:CY%(LC%)=J%
1210 GOTO 890
1220 CX2%=CX2%:CX1%=CX1%:VRY%=VRY%
1230 CLS #1:PRINT #1," I CHOOSE LINE NUMBER ";CX2%
1240 PRINT #1,"    AND COLUMN NUMBER ";CX1%
1250 LOCATE #1,1,4:PRINT #1,"THAT GIVES ME ";VRY%;" SQU
     ARE(S)"
1260 PLOT 270+(30*CX1%),70+(30*CX2%):DRAW 290+(30*CX1%)
     ,89+(30*CX2%),N%
1270 PLOT 290+(30*CX1%),70+(30*CX2%):DRAW 270+(30*CX1%)
     ,89+(30*CX2%),N%
1280 FOR T=1 TO 1000:NEXT T
1290 FOR K%=1 TO 8
1300 VR%=0:C3%=CX1%:C4%=CX2%
1310 C3%=C3%+C1%(K%):C4%=C4%+C2%(K%)
1320 IF C%(C3%,C4%)=Q% GOTO 1330 ELSE 1340
1330 VR%=VR%+1:GOTO 1310
1340 IF C%(C3%,C4%)=6 GOTO 1350 ELSE 1360
1350 NEXT K%:GOTO 1400
1360 IF VR%=0 GOTO 1350
1370 C3%=C3%-C1%(K%):C4%=C4%-C2%(K%)
1380 IF C%(C3%,C4%)=6 GOTO 1350
1390 C%(C3%,C4%)=N%:GOTO 1370
1400 C%(CX1%,CX2%)=N%
1410 GOSUB 2720
1420 GOSUB 1460
1430 IF PAS%=1 GOTO 1440 ELSE 1450
1440 CLS #1:PRINT #1,"     YOU MUST PASS":FOR T=1 TO 10
     00:NEXT T:GOTO 770
1450 GOTO 400
1460 FOR I%=1 TO 8
1470 FOR J%=1 TO 8
1480 FOR M%=0 TO 19 STEP 2
1490 Z%=270+(30*I%):H%=70+(30*J%):W%=H%+M%
1500 PLOT Z%,W%:DRAW Z%+20,W%,C%(I%,J%)
1510 NEXT M%:NEXT J%:NEXT I%
```

continua sulla prossima pagina

```
1520 X%=X%+1
1530 IF X%=61 GOTO 1550
1540 RETURN
1550 CQ%=0:CN%=0
1560 FOR I%=1 TO 8
1570 FOR J%=1 TO 8
1580 IF C%(I%,J%)=Q% THEN CQ%=CQ%+1
1590 IF C%(I%,J%)=N% THEN CN%=CN%+1
1600 NEXT J%:NEXT I%
1610 IF CQ% > CN% GOTO 1680
1620 IF CQ%=CN% GOTO 1630 ELSE 1650
1630 CLS #1:LOCATE #1,25,2:PRINT #1,"DEADLOCK"
1640 END
1650 CLS #1:LOCATE #1,5,1:PRINT #1,"YOU HAVE ";CQ%;" SQ
     UARES;I HAVE ";CN%
1660 LOCATE #1,11,3:PRINT #1,"I HAVE WON....!!!!"
1670 END
1680 CLS #1:LOCATE #1,5,1:PRINT #1,"YOU HAVE ";CQ%;" SQ
     UARES;I HAVE ";CN%
1690 LOCATE #1,5,3:PRINT #1,"WELL DONE. YOU HAVE WON !!
     "
1700 END
1710 IF C%(2,2)=Q% AND (C%(3,1)=N% OR C%(1,3)=N%) GOTO
     1720 ELSE 1730
1720 P%(3,1)=1:P%(1,3)=1
1730 IF C%(7,7)=Q% AND (C%(8,6)=N% OR C%(6,8)=N%) GOTO
     1740 ELSE 1750
1740 P%(8,6)=1:P%(6,8)=1
1750 IF C%(2,7)=Q% AND (C%(1,6)=N% OR C%(3,8)=N%) GOTO
     1760 ELSE 1770
1760 P%(1,6)=1:P%(3,8)=1
1770 IF C%(7,2)=Q% AND (C%(6,1)=N% OR C%(8,3)=N%) GOTO
     1780 ELSE 1790
1780 P%(6,1)=1:P%(8,3)=1
1790 IF D%=1 OR D%=8 OR E%=1 OR E%=8 GOTO 1820
1800 IF CX1%=1 OR CX1%=8 OR CX2%=1 OR CX2%=8 GOTO 1820
1810 RETURN
1820 FOR J%=1 TO 8 STEP 7
1830 FOR I%=2 TO 7
1840 IF C%(I%,J%)=N% GOTO 1850 ELSE 1860
1850 P%(I%+1,J%)=21:P%(I%-1,J%)=21
1860 IF C%(J%,I%)=N% GOTO 1870 ELSE 1880
1870 P%(J%,I%+1)=21:P%(J%,I%-1)=21
1880 NEXT I%
```

continua sulla prossima pagina

```
1890 FOR I%=2 TO 7
1900 IF C%(I%,J%)=Q% GOTO 1910 ELSE 1920
1910 P%(I%+1,J%)=2:P%(I%-1,J%)=2
1920 IF C%(J%,I%)=Q% GOTO 1930 ELSE 1940
1930 P%(J%,I%+1)=2:P%(J%,I%-1)=2
1940 NEXT I%:NEXT J%
1950 P%(1,2)=1:P%(1,7)=1:P%(2,1)=1:P%(7,1)=1
1960 P%(2,8)=1:P%(7,8)=1:P%(8,2)=1:P%(8,7)=1
1970  FOR I%=2 TO 7
1980 IF C%(1,I%-1)=Q% AND C%(1,I%+1)=Q% THEN P%(1,I%)=2
     5
1990 IF C%(8,I%-1)=Q% AND C%(8,I%+1)=Q% THEN P%(8,I%)=2
     5
2000 IF C%(I%-1,1)=Q% AND C%(I%+1,1)=Q% THEN P%(I%,1)=2
     5
2010 IF C%(I%-1,8)=Q% AND C%(I%+1,8)=Q% THEN P%(I%,8)=2
     5
2020 NEXT I%
2030 FOR J%=1 TO 8 STEP 7
2040 FOR I%=4 TO 8
2050 IF C%(J%,I%) <> N% GOTO 2140
2060 IC%=I%-1:IF C%(J%,IC%)=6 GOTO 2140
2070 IF C%(J%,IC%)=Q% GOTO 2080 ELSE 2090
2080 IC%=IC%-1:GOTO 2070
2090 IF C%(J%,IC%)=6 GOTO 2110
2100 GOTO 2140
2110 IF IC%=0 GOTO 2140
2120 IF C%(J%,I%+1)=Q% AND C%(J%,IC%-1)=6 GOTO 2140
2130 P%(J%,IC%)=26
2140 IF C%(I%,J%) <> N% GOTO 2230
2150 IC%=I%-1:IF C%(IC%,J%)=6 GOTO 2230
2160 IF C%(IC%,J%)=Q% GOTO 2170 ELSE 2180
2170 IC%=IC%-1:GOTO 2160
2180 IF C%(IC%,J%)=6 GOTO 2200
2190 GOTO 2230
2200 IF IC%=0 GOTO 2230
2210 IF C%(I%+1,J%)=Q% AND C%(IC%-1,J%)=6 GOTO 2230
2220 P%(IC%,J%)=26
2230 NEXT I%
2240 FOR I%=1 TO 5
2250 IF C%(J%,I%) <> N% GOTO 2340
2260 IC%=I%+1:IF C%(J%,IC%)=6 GOTO 2340
2270 IF C%(J%,IC%)=Q% GOTO 2280 ELSE 2290
2280 IC%=IC%+1:GOTO 2270
```

continua sulla prossima pagina

```
2290 IF C%(J%,IC%)=6 GOTO 2310
2300 GOTO 2340
2310 IF IC%=9 GOTO 2340
2320 IF C%(J%,I%-1)=Q% AND C%(J%,IC%+1)=6 GOTO 2340
2330 P%(J%,IC%)=26
2340 IF C%(I%,J%) <> N% GOTO 2430
2350 IC%=I%+1:IF C%(IC%,J%)=6 GOTO 2430
2360 IF C%(IC%,J%)=Q% GOTO 2370 ELSE 2380
2370 IC%=IC%+1:GOTO 2360
2380 IF C%(IC%,J%)=6 GOTO 2400
2390 GOTO 2430
2400 IF IC%=9 GOTO 2430
2410 IF C%(I%-1,J%)=Q% AND C%(IC%+1,J%)=6 GOTO 2430
2420 P%(IC%,J%)=26
2430 NEXT I%:NEXT J%
2440 IF C%(1,1)=N% GOTO 2450 ELSE 2460
2450 FOR I%=2 TO 6:P%(1,I%)=20:P%(I%,1)=20:NEXT I%
2460 IF C%(1,8)=N% GOTO 2470 ELSE 2480
2470 FOR I%=2 TO 6:P%(I%,8)=20:P%(1,9-I%)=20:NEXT I%
2480 IF C%(8,1)=N% GOTO 2490 ELSE 2500
2490 FOR I%=2 TO 6:P%(9-I%,1)=20:P%(8,I%)=20:NEXT I%
2500 IF C%(8,8)=N% GOTO 2510 ELSE 2520
2510 FOR I%=3 TO 7:P%(I%,8)=20:P%(8,I%)=20:NEXT I%
2520 IF C%(1,1) <> 6 THEN P%(2,2)=5
2530 IF C%(1,8) <> 6 THEN P%(2,7)=5
2540 IF C%(8,1) <> 6 THEN P%(7,2)=5
2550 IF C%(8,8) <> 6 THEN P%(7,7)=5
2560 P%(1,1)=30:P%(1,8)=30:P%(8,1)=30:P%(8,8)=30
2570 FOR I%=3 TO 6
2580 IF C%(1,I%)=N% THEN P%(2,I%)=4
2590 IF C%(8,I%)=N% THEN P%(7,I%)=4
2600 IF C%(I%,1)=N% THEN P%(I%,2)=4
2610 IF C%(I%,8)=N% THEN P%(I%,7)=4
2620 NEXT I%
2630 IF C%(7,1)=Q% AND C%(4,1)=N% AND C%(6,1)=6 AND C%(
     5,1)=6 THEN P%(6,1)=26
2640 IF C%(1,7)=Q% AND C%(1,4)=N% AND C%(1,6)=6 AND C%(
     1,5)=6 THEN P%(1,6)=26
2650 IF C%(2,1)=Q% AND C%(5,1)=N% AND C%(3,1)=6 AND C%(
     4,1)=6 THEN P%(3,1)=26
2660 IF C%(1,2)=Q% AND C%(1,5)=N% AND C%(1,3)=6 AND C%(
     1,4)=6 THEN P%(1,3)=26
2670 IF C%(8,2)=Q% AND C%(8,5)=N% AND C%(8,3)=6 AND C%(
     8,4)=6 THEN P%(8,3)=26
```

continua sulla prossima pagina

```
2680 IF C%(2,8)=Q% AND C%(5,8)=N% AND C%(3,8)=6 AND C%(
     4,8)=6 THEN P%(3,8)=26
2690 IF C%(8,7)=Q% AND C%(8,4)=N% AND C%(8,5)=6 AND C%(
     8,6)=6 THEN P%(8,6)=26
2700 IF C%(7,8)=Q% AND C%(4,8)=N% AND C%(5,8)=6 AND C%(
     6,8)=6 THEN P%(6,8)=26
2710 RETURN
2720 PAS%=0
2730 FOR I%=1 TO 8
2740 FOR J%=1 TO 8
2750 IF C%(I%,J%)=Q% GOTO 2780
2760 NEXT J%:NEXT I%
2770 PAS%=1:RETURN
2780 FOR K%=1 TO 8
2790 VR%=0:C3%=I%:C4%=J%
2800 C3%=C3%+C1%(K%):C4%=C4%+C2%(K%)
2810 IF C3% < 1 OR C3% > 8 GOTO 2820 ELSE 2830
2820 NEXT K%:GOTO 2760
2830 IF C4% < 1 OR C4% > 8 GOTO 2820 ELSE 2840
2840 IF C%(C3%,C4%)=N% GOTO 2850 ELSE 2860
2850 VR%=VR%+1:GOTO 2800
2860 IF C%(C3%,C4%)=Q% GOTO 2820 ELSE 2870
2870 IF VR% > 0 THEN RETURN
2880 GOTO 2820
2890 DATA 1,0,30,1,20,10,10,20,1,30,1,1,1,1,3
2900 DATA 3,3,3,1,1,0,1,20,3,5,5,5,5,3,20,-1,1,10,3,5
2910 DATA 0,0,5,3,10,-1,0,10,3,5,0,0,5,3,10,-1
2920 DATA -1,20,3,5,5,5,5,3,20,0,-1,1,1,3,3,3,3,1,1,1,-
     1,30,1,20,10,10,20,1,30
2930 DATA 263,100,263,120,270,130,255,130,255,130,255,1
     40,255,140,270,140
2940 DATA 270,140,270,150,270,150,255,150,255,160,270,1
     60,270,160,270,180
2950 DATA 270,180,255,180,270,170,255,170,270,190,270,2
     10,270,200,255,200
2960 DATA 255,200,255,210,255,220,270,220,270,220,270,2
     30,270,230,255,230
2970 DATA 255,230,255,240,255,240,270,240,255,250,270,2
     50,270,250,270,260
2980 DATA 270,260,255,260,255,250,255,270,270,280,270,3
     00,270,300,255,300
2990 DATA 255,310,255,330,255,330,270,330,270,330,270,3
     10,270,310,255,310
3000 DATA 255,320,270,320
```

continua sulla prossima pagina

```
3010 DATA 310,355,310,375,350,355,335,355,335,355,335,3
     65,335,365,350,365
3020 DATA 350,365,350,375,350,375,335,375,365,355,380,3
     55,380,355,380,375
3030 DATA 380,375,365,375,380,365,365,365,410,355,410,3
     75,410,365,395,365
3040 DATA 395,365,395,375,425,355,440,355,440,355,440,3
     65,440,365,425,365
3050 DATA 425,365,425,375,425,375,440,375,455,375,455,3
     55,455,355,470,355
3060 DATA 470,355,470,365,470,365,455,365,485,375,500,3
     75,500,375,500,355
3070 DATA 515,375,515,355,515,355,530,355,530,355,530,3
     75,530,375,515,375
3080 DATA 515,365,530,365
```

Raffles

Si irrompe nella casa di Sua Signoria per rubare il tesoro. Molti ostacoli sbarreranno il cammino, luci da accendere, il cane da guardia! Per un giocatore contro il computer. Tastiera o Joystick.

```
10 'RAFFLES by DAVID RADISIC
20 'copyright (c) AMSOFT 1985
30 '
40 MODE 0:INK 0,0:BORDER 0:INK 1,26:INK 2,15:INK 3,25
50 INK 4,14:INK 5,24,12:INK 6,0:INK 7,0:INK 8,0:PAPER #
   1,7
60 delay=200
70 DIM objx(5,20),objy(5,20),gemx(5,20),gemy(5,20)
80 GOSUB 380
90 GOSUB 720
100 pause=200:GOSUB 340
110 IF gems=0 THEN GOSUB 970
120 PEN 4
130 FOR i=10 TO 12
140 LOCATE 15,i:PRINT"SWAG";
150 NEXT
160 PAPER 0:CLS#2:PAPER 8
170 GOSUB 1170
180 GOSUB 1230
190 GOSUB 1370
200 GOSUB 1510
210 IF rm=0 THEN GOSUB 1900
220 IF dead=0 THEN 160
230 pause=100:GOSUB 340
240 PAPER 0:CLS:PEN 1
250 LOCATE 4,3:PRINT"Do you want";
260 LOCATE 4,5:PRINT"Another game";
270 PEN 5:LOCATE 9,7:PRINT"Y/N";
280 i$=UPPER$(INKEY$):IF i$<>"Y" AND i$<>"N" THEN 280
290 IF i$="N" THEN MODE 2:PEN 1:STOP
300 RUN
310 IF dog=1 THEN RETURN
320 dog=1:dogx=minx(rm):dogy=miny(rm)
330 RETURN
340 FOR loop=1 TO pause
350 FRAME
360 NEXT
370 RETURN
```

continua sulla prossima pagina

```
380 rm=1:xp=6:yp=4:man$=CHR$(224):dog=0:stolen=0
390 SYMBOL 240,8,8,8,8,8,8,8,8
400 SYMBOL 241,0,0,0,0,255,0,0,0
410 SYMBOL 242,0,0,0,0,15,8,8,8
420 SYMBOL 243,0,0,0,0,248,8,8,8
430 SYMBOL 244,8,8,8,8,248,0,0,0
440 SYMBOL 245,8,8,8,8,15,0,0,0
450 SYMBOL 246,8,12,13,14,12,12,8,8
460 SYMBOL 247,8,12,12,14,13,12,8,8
470 SYMBOL 248,8,24,88,56,24,24,8,8
480 SYMBOL 249,8,24,24,56,88,24,8,8
490 SYMBOL 250,0,0,255,129,129,129,255,0
500 SYMBOL 251,28,20,20,20,20,20,20,28
510 SYMBOL 252,0,0,255,255,255,255,255,0
520 SYMBOL 253,28,28,28,28,28,28,28,28
530 SYMBOL 255,195,165,60,126,90,60,36,24
540 ENT 1,12,-4,1
550 ENT -2,=1000,60,=3000,40
560 ENV 1,10,1,5,2,-4,1,2,-1,20
570 windw$(1)=STRING$(2,250):windw$(2)=CHR$(251)+CHR$(8
    )+CHR$(10)+CHR$(251)+CHR$(8)+CHR$(10)+CHR$(251)
580 door$(1)=STRING$(2,252):door$(2)=CHR$(253)+CHR$(8)+
    CHR$(10)+CHR$(253)+CHR$(8)+CHR$(10)+CHR$(253)
590 switch$(1,0)=CHR$(246):switch$(1,1)=CHR$(247)
600 switch$(2,0)=CHR$(248):switch$(2,1)=CHR$(249)
610 gem$=CHR$(144):obj$=CHR$(233):dog$=CHR$(255)
620 hit$=CHR$(246)+CHR$(248)+CHR$(247)+CHR$(249)+CHR$(2
    52)+CHR$(253)+CHR$(250)+CHR$(251)+gem$+obj$+dog$
630 RESTORE 3010
640 FOR i=1 TO 5
650 READ minx(i),miny(i),maxx(i),maxy(i)
660 READ dir(i,1),dir(i,2),dir(i,3),dir(i,4)
670 NEXT
680 WINDOW #1,minx(rm)-1,maxx(rm)+1,miny(rm)-1,maxy(rm)
    +1
690 WINDOW #2,1,14,1,25
700 CLS #1:PAPER #0,8
710 RETURN
720 ORIGIN 50,50
730 INK 6,24,12
740 RESTORE 3060
750 GOSUB 1280
760 LOCATE 3,20
770 PEN 5:PRINT">";
```

continua sulla prossima pagina

```
780 PEN 1:PRINT"Escape  Routes";
790 PEN 5:PRINT"<";:PEN 1
800 LOCATE 10,2:PRINT"IN";
810 pause=300:GOSUB 340
820 CLS:LOCATE 1,3:INK 6,0
830 PEN 1:PRINT man$;"  You The Thief":PRINT
840 PEN 2:PRINT LEFT$(door$(1),1);LEFT$(door$(2),1);" D
    oors":PRINT
850 PEN 3:PRINT switch$(1,0);switch$(2,0);" LightSwitch
     (OFF)"
860 PEN 3:PRINT switch$(1,1);switch$(2,1);" LightSwitch
     (ON)":PRINT
870 PEN 4:PRINT LEFT$(windw$(1),1);LEFT$(windw$(2),1);"
     Windows":PRINT
880 PEN 5:PRINT gem$;"  Precious Gems":PRINT
890 PAPER 1:PEN 0:PRINT obj$;"  Obstructions ":PEN 1:PA
    PER 0:PRINT
900 PEN 1:PRINT dog$;"  The Dog"
910 PEN 5:PRINT:PRINT:PRINT
920 PRINT"  Use Joystick":PRINT"   Or Cursor keys"
930 dummy=REMAIN(1)
940 AFTER delay*4,1 GOSUB 340
950 RETURN
960 '
970 'Generate Gems/obstacles
980 '
990 FOR room=1 TO 5
1000 gemr=INT(RND*8)+2:objr=INT(RND*10)+5
1010 minx=minx(room):miny=miny(room):maxx=maxx(room):ma
     xy=maxy(room)
1020 FOR i=1 TO gemr
1030 x=INT(RND*(maxx-minx+1))+minx
1040 y=INT(RND*(maxy-miny+1))+miny
1050 gemx(room,i)=x:gemy(room,i)=y
1060 gems=gems+1
1070 NEXT i
1080 FOR i=1 TO objr
1090 x=INT(RND*(maxx-minx+1))+minx
1100 y=INT(RND*(maxy-miny+1))+miny
1110 objx(room,i)=x:objy(room,i)=y
1120 NEXT i
1130 gems(room)=gemr:obj(room)=objr
1140 NEXT room
1150 CLS
```

continua sulla prossima pagina

```
1160 RETURN
1170 ON rm GOTO 1180,1190,1200,1210,1220
1180 RESTORE 2670:RETURN
1190 RESTORE 2740:RETURN
1200 RESTORE 2810:RETURN
1210 RESTORE 2880:RETURN
1220 RESTORE 2960:RETURN
1230 PAPER 0:READ rm$:PAPER 8
1240 WINDOW #1,minx(rm)-1,maxx(rm)+1,miny(rm)-1,maxy(rm
     )+1:CLS #1
1250 PEN 1:LOCATE 1,1:PRINT SPACE$(19);
1260 LOCATE 1,1:PRINT "Room :";rm$;
1270 IF lights(rm) THEN INK 7,10:INK 8,10 ELSE INK 7,0:
     INK 8,0
1280 READ a$:IF a$="END" THEN RETURN
1290 IF a$="D" THEN 2180
1300 IF a$="W" THEN 2260
1310 IF a$="L" THEN GRAPHICS PEN 1:GOTO 2340
1320 IF a$="S" THEN 2420
1330 IF a$="F" THEN GRAPHICS PEN 6:GOTO 2340
1340 PRINT"***ERROR ***";
1350 STOP
1360 '
1370 'Display gems/objects
1380 '
1390 PEN 6
1400 FOR i=1 TO obj(rm)
1410 LOCATE objx(rm,i),objy(rm,i)
1420 PRINT obj$;
1430 NEXT
1440 PEN 5
1450 FOR i=1 TO gems(rm)
1460 LOCATE gemx(rm,i),gemy(rm,i)
1470 PRINT gem$;
1480 NEXT
1490 PEN 1:LOCATE xp,yp:PRINT man$;
1500 RETURN
1510 xf=0:yf=0:PEN 1
1520 IF INKEY(0)<>-1 OR INKEY(72)<>-1 THEN yf=-1
1530 IF INKEY(2)<>-1 OR INKEY(73)<>-1 THEN yf=1
1540 IF INKEY(8)<>-1 OR INKEY(74)<>-1 THEN xf=-1
1550 IF INKEY(1)<>-1 OR INKEY(75)<>-1 THEN xf=1
1560 IF xf=0 AND yf=0 THEN 1630
1570 LOCATE xp+xf,yp+yf:ht$=COPYCHR$(#0)
```

continua sulla prossima pagina

```
1580 IF ASC(ht$)>239 AND ASC(ht$)<246 THEN 1510
1590 IF ht$<>" " THEN 1660
1600 LOCATE xp,yp:PRINT " ";
1610 PAPER 0:LOCATE 15,5:PRINT"      ";:PAPER 8
1620 xp=xp+xf:yp=yp+yf
1630 LOCATE xp,yp:PRINT man$;
1640 IF dog>0 THEN dog=dog MOD 2+1:IF dog=2 THEN 2550
1650 GOTO 1510
1660 hit=INSTR(hit$,ht$):char=ASC(MID$(hit$,hit,1))
1670 ON hit GOTO 1690,1690,1690,1690,1750,1750,1850,190
     0,1970,2090,2650
1680 GOTO 1600
1690 IF hit>2 AND hit<5 THEN char=char-1
1700 IF hit<3 THEN char=char+1
1710 PEN 3:LOCATE xp+xf,yp+yf:PRINT CHR$(char);
1720 lights(rm)=lights(rm) XOR 1
1730 IF lights(rm) THEN INK 7,10:INK 8,10 ELSE INK 7,0:
     INK 8,0
1740 GOTO 1510
1750 IF xf<>0 AND yf<>0 THEN 1630
1760 IF xf<0 THEN dir=4 ELSE IF xf>0 THEN dir=3
1770 IF yf<0 THEN dir=1 ELSE IF yf>0 THEN dir=2
1780 IF dir(rm,dir)=-1 THEN 1630 ELSE rm=dir(rm,dir)
1790 IF dog>0 THEN GOSUB 310
1800 IF dir=1 THEN xp=6:yp=maxy(rm)
1810 IF dir=2 THEN xp=6:yp=miny(rm)
1820 IF dir=3 THEN xp=minx(rm):yp=13
1830 IF dir=4 THEN xp=maxx(rm):yp=13
1840 RETURN
1850 IF xp>5 AND xp<8 THEN 1880
1860 IF xp<6 THEN dir=4 ELSE dir=3
1870 GOTO 1780
1880 IF yp>13 THEN dir=2 ELSE dir=1
1890 GOTO 1780
1900 PAPER 0:CLS:PEN 1
1910 LOCATE 3,3:PRINT"You have escaped";
1920 LOCATE 9,5:PRINT"with";
1930 IF gems=stolen THEN LOCATE 8,7:PRINT"All"; ELSE LO
     CATE 9,7
1940 PRINT USING " ##";stolen;
1950 PEN 5:LOCATE 9,9:PRINT"Gems";
1960 dead=1:RETURN
1970 LOCATE xp,yp:PRINT" ";:xp=xp+xf:yp=yp+yf
```

continua sulla prossima pagina

```
1980 i=0
1990 i=i+1
2000 IF i>gems(rm) THEN 1510
2010 IF gemx(rm,i)<>xp OR gemy(rm,i)<>yp THEN 1990
2020 IF i=gems(rm) THEN 2050
2030 gemx(rm,i)=gemx(rm,gems(rm))
2040 gemy(rm,i)=gemy(rm,gems(rm))
2050 gems(rm)=gems(rm)-1:stolen=stolen+1
2060 MOVE 400,150+(stolen*2),1,1:DRAW 520,150+(stolen*2
     ),1,1
2070 SOUND 129,248,10,12,0,1
2080 GOTO 1980
2090 noise=INT(RND*15)
2100 SOUND 1,3000,10,noise,0,0,10
2110 PAPER 0:LOCATE 15,5:PRINT"Crash ";:PAPER 8
2120 IF noise<10 OR delay=50 THEN 1630
2130 delay=delay-50
2140 dummy=REMAIN(1)
2150 AFTER delay*4,1 GOSUB 310
2160 GOTO 1630
2170 '
2180 'Draw doors
2190 '
2200 READ no,dr$
2210 IF dr$="V" THEN dr=2 ELSE dr=1
2220 PEN 2
2230 pic$=door$(dr):GOSUB 2500
2240 GOTO 1280
2250 '
2260 'Draw windows
2270 '
2280 READ no,wi$
2290 IF wi$="V" THEN wi=2 ELSE wi=1
2300 PEN 4
2310 pic$=windw$(wi):GOSUB 2500
2320 GOTO 1280
2330 '
2340 ' Draw lines
2350 '
2360 READ x1,y1,x2,y2
2370 MOVE x1,y1,,0
2380 DRAW x1,y2,,0:DRAW x2,y2,,0
2390 DRAW x2,y1,,0:DRAW x1,y1,,0
2400 GOTO 1280
```

continua sulla prossima pagina

```
2410 '
2420 'Draw switches
2430 '
2440 READ no,sw$
2450 IF sw$="L" THEN sw=1 ELSE sw=2
2460 PEN 3
2470 pic$=switch$(sw,0):GOSUB 2500
2480 GOTO 1280
2490 '
2500 'Print char
2510 '
2520 READ x,y:LOCATE x,y:PRINT pic$;
2530 no=no-1:IF no>0 THEN 2520
2540 RETURN
2550 PEN 1:LOCATE dogx,dogy:PRINT" ";
2560 man$=CHR$(225)
2570 IF (dogx=xp AND dogy=yp) OR (dogx=xp+xf AND dogy=y
     p+yf) THEN 2650
2580 IF dogx<xp THEN dogx=dogx+1
2590 IF dogx>xp THEN dogx=dogx-1
2600 IF dogy<yp THEN dogy=dogy+1
2610 IF dogy>yp THEN dogy=dogy-1
2620 LOCATE dogx,dogy:PRINT dog$;
2630 SOUND 1,0,RND*40,10,1,2,31
2640 GOTO 1510
2650 PRINT"SNAP";
2660 dead=1:RETURN
2670 DATA Hallway
2680 DATA L,64,308,226,4
2690 DATA D,2,H,6,3,6,22
2700 DATA D,2,V,4,12,9,11
2710 DATA S,1,L,4,11
2720 DATA S,1,R,9,14
2730 DATA END
2740 DATA Lounge
2750 DATA L,2,308,258,4
2760 DATA D,1,V,10,12
2770 DATA W,1,H,6,3
2780 DATA W,1,V,2,12
2790 DATA S,2,R,10,11,10,15
2800 DATA END
2810 DATA Dining room
2820 DATA L,2,308,258,4
2830 DATA W,1,V,10,12
```

continua sulla prossima pagina

```
2840 DATA W,1,H,6,3
2850 DATA D,1,V,2,12
2860 DATA S,2,L,2,11,2,15
2870 DATA END
2880 DATA Kitchen
2890 DATA L,2,276,384,4
2900 DATA D,2,H,6,5,6,22
2910 DATA W,1,H,10,22
2920 DATA W,1,V,14,13
2930 DATA D,1,V,2,13
2940 DATA S,1,L,2,16
2950 DATA END
2960 DATA Pantry
2970 DATA L,2,276,256,4
2980 DATA D,1,V,10,12
2990 DATA S,1,R,10,11
3000 DATA END
3010 DATA 5,4,8,21,0,4,3,2
3020 DATA 3,4,9,21,-1,-1,1,-1
3030 DATA 3,4,9,21,-1,-1,-1,1
3040 DATA 3,6,13,21,1,0,-1,5
3050 DATA 3,6,9,21,-1,-1,4,-1
3060 DATA L,64,308,480,100
3070 DATA F,250,98,294,102
3080 DATA F,250,306,294,310
3090 DATA F,390,94,430,106
3100 DATA F,390,302,430,314
3110 DATA F,474,240,488,270
3120 DATA F,474,124,488,154
3130 DATA F,58,240,72,270
3140 DATA L,226,308,322,180
3150 DATA L,160,180,480,100
3160 DATA L,64,180,160,100
3170 DATA END
```

Appendice 4
Indice analitico

I numeri indicati si riferiscono rispettivamente al Capitolo ed alla pagina in cui si trovano i riferimenti alla voce.

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z

AMSTRAD

464 PLUS 6128 PLUS